



Dirk Louis
Christian Wenz



Dynamic Web-Publishing

Webseiten
der 4. Generation

HTML, JavaScript, CSS,
ASP, Perl, CGI, Java,
PHP

Alle Projekte auf der
Buch-CD



Das komplette Buch im
HTML-Format; Quellen für alle
Programmbeispiele, wertvolle Tools
für Web-Publisher

Vorwort

Tag 1 Von der Idee zum eigenen Web

[1.1 Idee und Design](#)

[1.2 Aufsetzen des Codes - der HTML-Editor](#)

[1.3 Webseiten im Browser testen](#)

[1.4 Testen auf einem lokalen Webserver](#)

[1.5 Der Schritt in die Öffentlichkeit -
das Hochladen auf den Internet-Webserver](#)

[1.6 Die richtige Wahl des Internet Service Providers](#)

[1.7 Zusammenfassung](#)

[1.8 Fragen und Antworten](#)

[1.9 Workshop](#)

Tag 2 Am Anfang war... HTML

[2.1 Was sind Markups?](#)

[2.2 Das HTML-Grundgerüst](#)

[2.3 Text und Überschriften](#)

[2.4 Listen](#)

[2.5 Bilder](#)

[2.6 Hyperlinks](#)

[2.7 Sonstige Tags](#)

[2.8 Inline-Stile, Farben, Schriften und Seitenhintergrund](#)

[2.9 Zusammenfassung](#)

[2.10 Fragen und Antworten](#)

[2.11 Workshop](#)

Tag 3 Tabellen, Frames und Formulare

[3.1 Tabellen](#)

[3.2 Frames](#)

[3.3 Inline-Frames](#)

[3.4 Steuerelemente und Formulare](#)

[3.5 Zusammenfassung](#)

[3.6 Fragen und Antworten](#)

[3.7 Workshop](#)

Tag 4 Formatieren mit Stylesheets

[4.1 Wozu braucht man Stylesheets?](#)

[4.2 Stylesheets definieren und zuweisen I](#)

[4.3 Die einzelnen Stileigenschaften](#)

[4.4 Stylesheets definieren und zuweisen II](#)

[4.5 Vererbung und Cascading](#)

[4.6 CSS-Stylesheets und die Browser](#)

[4.7 Zusammenfassung](#)

[4.8 Fragen und Antworten](#)

[4.9 Workshop](#)

Tag 5 Positionieren mit Stylesheets

[5.1 Vor- und Nachteile der traditionellen Positionierung](#)

[5.2 Grundlagen der CSS-Positionierung](#)

[5.3 Die Positionierungsattribute](#)

[5.4 Freies Webdesign dank CSS-Positionierung](#)

[5.5 Sichtbarkeit und Clipping](#)

[5.6 Textfluss um Bilder](#)

[5.7 Zusammenfassung](#)

[5.8 Fragen und Antworten](#)

[5.9 Workshop](#)

Tag 6 Fortgeschrittenes und dynamisches HTML

[6.1 Multimediale Objekte einfügen](#)

[6.2 Quickinfos](#)

[6.3 Metainformationen und Header-Tags](#)

[6.4 Suchmaschinen unterstützen](#)

[6.5 Zusammenfassung](#)

[6.6 Fragen und Antworten](#)

[6.7 Workshop](#)

Tag 7 Ein gelungener Webauftritt

[7.1 Mit einfachem Design anfangen](#)

[7.2 Inspiration finden](#)

[7.3 Todsünden des Webdesigns](#)

[7.4 Workshop](#)

Tag 8 Wozu JavaScript?

[8.1 Was ist JavaScript?](#)

[8.2 JavaScript-Code in Webseiten einbinden](#)

[8.3 Was kann man mit JavaScript anfangen -
ein Schnupperkurs](#)

[8.4 Fehler im JavaScript-Code finden](#)

[8.5 Zusammenfassung](#)

[8.6 Fragen und Antworten](#)

[8.7 Workshop](#)

Tag 9 JavaScript-Grundkurs

[9.1 Variablen, Konstanten und Datentypen](#)

[9.2 Operatoren und Ausdrücke](#)

[9.3 Steuerung des Programmablaufs](#)

[9.4 Funktionen](#)

[9.5 Klassen und Objekte](#)

[9.6 Vordefinierte Klassen und Objekte](#)

[9.7 Zusammenfassung](#)

[9.8 Fragen und Antworten](#)

[9.9 Workshop](#)

Tag 10 Mit JavaScript auf HTML-Elemente zugreifen

[10.1 Ereignisse](#)

[10.2 Das event-Objekt](#)

[10.3 Erweiterte Möglichkeiten zum Abfangen von Ereignissen](#)

[10.4 Das DOM-Modell](#)

[10.5 Browserunabhängige Lösungen](#)

[10.6 Rollover-Effekte für Schalter](#)

[10.7 Dynamisch aufklappbare Listen](#)

[10.8 Zusammenfassung](#)

[10.9 Fragen und Antworten](#)

[10.10 Workshop](#)

Tag 11 JavaScript und Formulare

[11.1 Zugriff und Ereignisverarbeitung für Steuerelemente und Formulare](#)

[11.2 Benutzeroberflächen](#)

[11.3 Formulareingaben](#)

[11.4 Zusammenfassung](#)

[11.5 Fragen und Antworten](#)

[11.6 Workshop](#)

Tag 12 JavaScript und Frames

[12.1 Frame-Navigation mit JavaScript](#)

[12.2 JavaScript-unterstützte Frame-Hyperlinks](#)

[12.3 Zugriff auf HTML-Elemente in Frames](#)

[12.4 Zusammenfassung](#)

[12.5 Fragen und Antworten](#)

[12.6 Workshop](#)

Tag 13 Cookies

[13.1 Was sind Cookies?](#)

[13.2 Cookies mit JavaScript setzen und abfragen](#)

[13.3 Cookies löschen](#)

[13.4 Einstellungen einer Seite speichern](#)

[13.5 Cookies, Virtuelle Warenkörbe und CGI](#)

[13.6 Zusammenfassung](#)

[13.7 Fragen und Antworten](#)

[13.8 Workshop](#)

Tag 14 Erste serverseitige Techniken

[14.1 Serverseitiges JavaScript](#)

[14.2 Server Side Includes](#)

[14.3 Zusammenfassung](#)

[14.4 Fragen und Antworten](#)

[14.5 Workshop](#)

Tag 15 Java-Applets

[15.1 Das Applet-Konzept](#)

[15.2 Eigene Applets erstellen](#)

[15.3 Parameter an Applets übergeben](#)

[15.4 Grafikausgaben in Applets](#)

[15.5 Auf Mausclicks reagieren](#)

[15.6 Wie geht es weiter?](#)

[15.7 Zusammenfassung](#)

[15.8 Fragen und Antworten](#)

[15.9 Workshop](#)

Tag 16 Animationen

[16.1 Animationen](#)

[16.2 GIF-Animationen](#)

[16.3 JavaScript-DHTML-Animationen](#)

[16.4 Java-Animationen](#)

[16.5 Flash-Animationen](#)

[16.6 Zusammenfassung](#)

[16.7 Fragen und Antworten](#)

[16.8 Workshop](#)

Tag 17 CGI und Perl

[17.1 Wie funktioniert CGI?](#)

[17.2 Perl](#)

[17.3 Webseiten dynamisch erzeugen](#)

[17.4 Serverseitige ImageMaps](#)

[17.5 Eigene Daten an CGI-Programme schicken](#)

[17.6 Formulare](#)

[17.7 Verborgene Formularfelder](#)

[17.8 Gästebücher](#)

[17.9 Zusammenfassung](#)

[17.10 Fragen und Antworten](#)

[17.11 Workshop](#)

Tag 18 Active Server Pages

[18.1 Wie funktioniert eigentlich ASP?](#)

[18.2 Spracheinführung](#)

[18.3 Datumswerte](#)

[18.4 Auf Formulare zugreifen](#)

[18.5 Mit Dateien arbeiten](#)

[18.6 Cookies](#)

[18.7 Fragen und Antworten](#)

[18.8 Workshop](#)

Tag 19 PHP - Einführung

[19.1 Was ist eigentlich PHP?](#)

[19.2 Spracheinführung](#)

[19.3 Datumswerte](#)

[19.4 Auf Formulare zugreifen](#)

[19.5 Mit Dateien arbeiten](#)

[19.6 Cookies](#)

[19.7 Fragen und Antworten](#)

[19.8 Workshop](#)

Tag 20 Datenbankbindung

[20.1 Was ist eine Datenbank?](#)

[20.2 SQL](#)

[20.3 ODBC](#)

[20.4 Datenbanken mit ASP](#)

[20.5 Datenbanken mit PHP](#)

[20.6 Fragen und Antworten](#)

[20.7 Workshop](#)

Tag 21 XML und XHTML

[21.1 XML](#)

[21.2 XHTML](#)

[21.3 Zusammenfassung](#)

[21.4 Fragen und Antworten](#)

[21.5 Workshop](#)

Anhang A [Einrichtung lokaler Webserver](#) 861

[A.1 Der Apache-Server](#)

[A.2 Der PWS- und der IIS-Server](#)

[A.3 Der OmniHTTPd-Server](#)

Anhang B [Sonderzeichen](#) 881

Anhang C [HTML-Elemente](#) 887

Anhang D [Lösungen](#) 891

[D.1 Tag 1: Von der Idee zum eigenen Web](#)

[D.2 Tag 2: Am Anfang war... HTML](#)
[D.3 Tag 3: Tabellen, Frames und Formulare](#)
[D.4 Tag 4: Formatieren mit Stylesheets](#)
[D.5 Tag 5: Positionieren mit Stylesheets](#)
[D.6 Tag 6: Fortgeschrittenes und dynamisches HTML](#)
[D.7 Tag 7: Ein gelungener Webauftritt](#)
[D.8 Tag 8: Wozu JavaScript?](#)
[D.9 Tag 9: JavaScript-Grundkurs](#)
[D.10 Tag 10: Mit JavaScript auf HTML-Elemente zugreifen](#)
[D.11 Tag 11: JavaScript und Formulare](#)
[D.12 Tag 12: JavaScript und Frames](#)
[D.13 Tag 13: Cookies](#)
[D.14 Tag 14: Erste serverseitige Techniken](#)
[D.15 Tag 15: Java-Applets](#)
[D.16 Tag 16: Animationen](#)
[D.17 Tag 17: CGI und Perl](#)
[D.18 Tag 18: Microsoft präsentiert... ASP](#)
[D.19 Tag 19: PHP, der Shooting-Star](#)
[D.20 Tag 20: Datenbankbindung](#)
[D.21 Tag 21: XML und XHTML](#)

Anhang E [Adressen](#) 945

[E.1 HTML](#)
[E.2 JavaScript](#)
[E.3 Java](#)
[E.4 Perl](#)
[E.5 CGI](#)
[E.6 ASP](#)
[E.7 PHP](#)
[E.8 XML](#)
[E.9 Software](#)

Anhang F [Die CD zum Buch](#) 949

Stichwortverzeichnis

Stichwortverzeichnis

Symbols

#config (SSI)

- [Erste serverseitige Techniken](#)

#echo (SSI)

- [Erste serverseitige Techniken](#)
- [Erste serverseitige Techniken](#)
- [Erste serverseitige Techniken](#)

#exec (SSI)

- [Erste serverseitige Techniken](#)

#lastmod (SSI)

- [Erste serverseitige Techniken](#)

#filesize (SSI)

- [Erste serverseitige Techniken](#)

#include (SSI)

- [Erste serverseitige Techniken](#)
- [Erste serverseitige Techniken](#)

(JavaScript-Array)

- [JavaScript und Formulare](#)
- [JavaScript und Frames](#)

<!ATTLIST>

- [XML und XHTML](#)

<!DOCTYPE>

- [Am Anfang war... HTML](#)
- [Tabellen, Frames und Formulare](#)
- [XML und XHTML](#)

<%

- [Active Server Pages](#)

<?php

- [PHP - Einführung](#)

<a>

- [Am Anfang war... HTML](#)

<abbr>

- [Am Anfang war... HTML](#)

<acronym>

- [Am Anfang war... HTML](#)

<address>

- [Am Anfang war... HTML](#)

<applet>

- [Java-Applets](#)

<area />

- [Am Anfang war... HTML](#)

- [Am Anfang war... HTML](#)

<base />

- [Fortgeschrittenes und dynamisches HTML](#)

<big>

- [Am Anfang war... HTML](#)

<blockquote>

- [Am Anfang war... HTML](#)

<body>

- [Am Anfang war... HTML](#)

**
**

- [Am Anfang war... HTML](#)

<caption>

- [Tabellen, Frames und Formulare](#)

<cite>

- [Am Anfang war... HTML](#)

<code>

- [Am Anfang war... HTML](#)

<col />

- [Tabellen, Frames und Formulare](#)

<colgroup>

- [Tabellen, Frames und Formulare](#)

<dd>

- [Am Anfang war... HTML](#)

- [Am Anfang war... HTML](#)

<dfn>

- [Am Anfang war... HTML](#)

<div>

- [Am Anfang war... HTML](#)

- [Positionieren mit Stylesheets](#)

<dl>

- [Am Anfang war... HTML](#)

<dt>

- [Am Anfang war... HTML](#)

- [Am Anfang war... HTML](#)

<embed>

- [Fortgeschrittenes und dynamisches HTML](#)

- [Am Anfang war... HTML](#)
- [Formatieren mit Stylesheets](#)

<form>

- [Tabellen, Frames und Formulare](#)

<frame />

- [Tabellen, Frames und Formulare](#)

<frameset>

- [Tabellen, Frames und Formulare](#)
- [Tabellen, Frames und Formulare](#)

<h1>

- [Am Anfang war... HTML](#)

<hr />

- [Am Anfang war... HTML](#)

<i>

- [Am Anfang war... HTML](#)

<iframe>

- [Tabellen, Frames und Formulare](#)

- [Am Anfang war... HTML](#)

<input />

- [Tabellen, Frames und Formulare](#)

<ins>

- [Am Anfang war... HTML](#)

<kbd>

- [Am Anfang war... HTML](#)

<label>

- [Tabellen, Frames und Formulare](#)

<layer>

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

- [Am Anfang war... HTML](#)

<link />

- [Formatieren mit Stylesheets](#)
- [Fortgeschrittenes und dynamisches HTML](#)

<map>

- [Am Anfang war... HTML](#)

<marquee>

- [Am Anfang war... HTML](#)

<meta />

- [Fortgeschrittenes und dynamisches HTML](#)

<noscript>

- [Wozu JavaScript?](#)

<object>

- [Fortgeschrittenes und dynamisches HTML](#)
- [Java-Applets](#)

- [Am Anfang war... HTML](#)

<option>

- [Tabellen, Frames und Formulare](#)

<p>

- [Am Anfang war... HTML](#)

<param>

- [Java-Applets](#)

<pre>

- [Am Anfang war... HTML](#)

<q>

- [Am Anfang war... HTML](#)

<samp>

- [Am Anfang war... HTML](#)

<script>-

- [Wozu JavaScript?](#)

<select>

- [Tabellen, Frames und Formulare](#)

<small>

- [Am Anfang war... HTML](#)

- [Am Anfang war... HTML](#)

- [Am Anfang war... HTML](#)

<style>

- [Formatieren mit Stylesheets](#)

<sub>

- [Am Anfang war... HTML](#)

<sup>

- [Am Anfang war... HTML](#)

<table>

- [Tabellen, Frames und Formulare](#)

<tbody>

- [Tabellen, Frames und Formulare](#)

<td>

- [Tabellen, Frames und Formulare](#)

<textarea>

- [Tabellen, Frames und Formulare](#)

<tfoot>

- [Tabellen, Frames und Formulare](#)

<th>

- [Tabellen, Frames und Formulare](#)

<thead>

- [Tabellen, Frames und Formulare](#)

<title>

- [Am Anfang war... HTML](#)

<tr>

- [Tabellen, Frames und Formulare](#)

<tt>

- [Am Anfang war... HTML](#)

- [Am Anfang war... HTML](#)

<var>

- [Am Anfang war... HTML](#)

<xsl>

- [XML und XHTML](#)

A

Absätze

- [Am Anfang war... HTML](#)

Active Server Pages

-

[Active Server Pages](#)

' (Kommentar)

■

[Active Server Pages](#)

<% %>

- [Active Server Pages](#)

—

- [Active Server Pages](#)

ADODB.Connection

- [Datenbankanbindung](#)

Arrays

- [Active Server Pages](#)

asp.dll

- [Active Server Pages](#)

Cookies

- [Active Server Pages](#)

Date

- [Active Server Pages](#)

Dateien

- [Active Server Pages](#)

Datenbankzugriff

- [Datenbankanbindung](#)

Datumswerte

- [Active Server Pages](#)

Day

- [Active Server Pages](#)

Dim

- [Active Server Pages](#)

Else-Anweisung

- [Active Server Pages](#)

Elseif-Anweisung

- [Active Server Pages](#)

Formulare

- [Active Server Pages](#)

Gästebuch

- [Datenbankanbindung](#)

Groß- und Kleinschreibung

- [Active Server Pages](#)

Hintergrundfarbe speichern

- [Active Server Pages](#)

Hour

- [Active Server Pages](#)

If-Anweisung

- [Active Server Pages](#)

InStr

- [Active Server Pages](#)

InStrRev

- [Active Server Pages](#)

IP-Adresse

- [Active Server Pages](#)

JScript

- [Active Server Pages](#)

Kommentare

- [Active Server Pages](#)

Left

- [Active Server Pages](#)

Len

- [Active Server Pages](#)

- [Active Server Pages](#)

Mid

- [Active Server Pages](#)

-

[Active Server Pages](#)

Minute

- [Active Server Pages](#)

Month

- [Active Server Pages](#)

Namensgebung

- [Active Server Pages](#)

Now

- [Active Server Pages](#)

Operatoren

Option Explicit

- [Active Server Pages](#)

Programmiersprache

- [Active Server Pages](#)

ReDim

- [Active Server Pages](#)

Rem

- [Active Server Pages](#)

Replace

- [Active Server Pages](#)
- [Datenbankanbindung](#)

Request.Cookies

- [Active Server Pages](#)

Request.Form

- [Active Server Pages](#)

Request.QueryString

- [Active Server Pages](#)

Request.ServerVariables

- [Active Server Pages](#)

Response.Cookies

- [Active Server Pages](#)

Response.Redirect

- [Active Server Pages](#)

Response.Write

- [Active Server Pages](#)

Right

- [Active Server Pages](#)

Schleifen

- [Active Server Pages](#)

Scripting.FileSystemObject

- [Active Server Pages](#)

Second

- [Active Server Pages](#)

Select Case

- [Active Server Pages](#)

Tagesdatum

- [Active Server Pages](#)

Text ausgeben

- [Active Server Pages](#)
- [Active Server Pages](#)

UBound

- [Active Server Pages](#)

Umleiten

- [Active Server Pages](#)

Variablen deklarieren

- [Active Server Pages](#)

VBScript

- [Active Server Pages](#)

vbTab

- [Active Server Pages](#)

Webserver

- [Active Server Pages](#)

Weekday

- [Active Server Pages](#)

- [Active Server Pages](#)

Wochentag

- [Active Server Pages](#)

Year

- [Active Server Pages](#)

Zuweisung

- [Active Server Pages](#)

addEventListener() (JavaScript-Methode)

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Adobe Photoshop

- [Von der Idee zum eigenen Web](#)

Adressen

- [HTML](#)

alert (JavaScript-Methode)

- [Wozu JavaScript?](#)

Animationen

- [Animationen](#)

Bilderwechsler

- [Animationen](#)
- [Animationen](#)

DHTML

- [Animationen](#)
- [Animationen](#)

dynamische Listen

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Flash

- [Animationen](#)

GIF-Animationen

- [Animationen](#)

Grundkonzepte

- [Animationen](#)

in Webseiten einbinden

- [Animationen](#)

Java

- [Animationen](#)

JavaScript-Animationen

- [Animationen](#)

Schalftflächen

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Sprite-Animationen

- [Animationen](#)

- [Animationen](#)

Zeichentrickanimationen

- [Animationen](#)

- [Animationen](#)

Antialiasing

- [Am Anfang war... HTML](#)

Anweisungen

- [JavaScript- Grundkurs](#)

Apache-Server

- [Von der Idee zum eigenen Web](#)

Applets

- [Java-Applets](#)

Argumente

- [JavaScript- Grundkurs](#)

Array (JavaScript-Klasse)

- [JavaScript- Grundkurs](#)

Arrays

- [JavaScript- Grundkurs](#)

ASP

- [Active Server Pages](#)

Attribute

- [Am Anfang war... HTML](#)

Aufzählungssymbole

- [Am Anfang war... HTML](#)

Ausdrücke

- [JavaScript- Grundkurs](#)

Automatische Aktualisierung

- [Fortgeschrittenes und dynamisches HTML](#)

Automatische Weiterschaltung

- [Fortgeschrittenes und dynamisches HTML](#)

B

Basisadresse

- [Fortgeschrittenes und dynamisches HTML](#)

Benutzereingaben

- [JavaScript- Grundkurs](#)

Benutzeroberflächen

- [JavaScript und Formulare](#)

Bilder

- [Am Anfang war... HTML](#)

Abmaße

- [Am Anfang war... HTML](#)

Adobe Photoshop

- [Von der Idee zum eigenen Web](#)

als Hyperlinks

- [Am Anfang war... HTML](#)

Antialiasing

- [Am Anfang war... HTML](#)

Bildformate

- [Am Anfang war... HTML](#)

Bildqualität

- [Am Anfang war... HTML](#)

blinde GIFs

- [Formatieren mit Stylesheets](#)
- [Positionieren mit Stylesheets](#)

einscannen

- [Am Anfang war... HTML](#)

Farbtiefe

- [Am Anfang war... HTML](#)

frei positionieren

- [Positionieren mit Stylesheets](#)

Hintergrundbilder

- [Am Anfang war... HTML](#)
- [Am Anfang war... HTML](#)

ImageMaps

- [Am Anfang war... HTML](#)
- [CGI und Perl](#)

Interlacing

- [Am Anfang war... HTML](#)

Klickereignisse mit JavaScript bearbeiten

- [Wozu JavaScript?](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Palette

- [Am Anfang war... HTML](#)

Positionierung

- [Am Anfang war... HTML](#)

Schaltflächen

- [Am Anfang war... HTML](#)

Skalierung

- [Am Anfang war... HTML](#)

Textfluss um Bilder

- [Positionieren mit Stylesheets](#)

Thumbnails

- [Am Anfang war... HTML](#)

Transparenz

- [Am Anfang war... HTML](#)

zerteilen und zusammenfügen

- [Am Anfang war... HTML](#)

Bildergalerien

- [Tabellen, Frames und Formulare](#)

Bildlaufleisten, in Frames

- [Tabellen, Frames und Formulare](#)

Binärcodierung

- [JavaScript- Grundkurs](#)

Body-Abschnitt

- [Am Anfang war... HTML](#)

Boolesche Attribute

- [Am Anfang war... HTML](#)
- [XML und XHTML](#)

Browser

Abspielprogramme

- [Fortgeschrittenes und dynamisches HTML](#)

Cookies

- [Cookies](#)

DHTML

- [Animationen](#)

DOM-Modell

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

event-Objekt

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Frames

- [Tabellen, Frames und Formulare](#)
- [Tabellen, Frames und Formulare](#)

globale Ereignisbearbeitung

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

HTML-Elemente positionieren

- [Positionieren mit Stylesheets](#)

Java-Applets

- [Java-Applets](#)

JavaScript

- [Wozu JavaScript?](#)

Stylesheets

- [Formatieren mit Stylesheets](#)
- [Formatieren mit Stylesheets](#)

Webseiten neu laden

- [Von der Idee zum eigenen Web](#)

Webseiten testen

- [Von der Idee zum eigenen Web](#)

Zeilenumbrüche erzwingen

- [Am Anfang war... HTML](#)

Bubbling-Up

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Bytecode

- [Java-Applets](#)

C

Caching

- [Fortgeschrittenes und dynamisches HTML](#)

Call by value

- [JavaScript- Grundkurs](#)

captureEvents() (proprietäre JavaScript-Methode)

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Cascading (Stylesheets)

- [Formatieren mit Stylesheets](#)

CGI

- [CGI und Perl](#)

Codierung von Browserdaten

- [CGI und Perl](#)

Content-type-Header

- [CGI und Perl](#)

Cookies

- [Cookies](#)

Daten an den URL eines CGI-Programms anhängen

- [CGI und Perl](#)

Datenübertragung

- [CGI und Perl](#)

Einsatzgebiete

- [CGI und Perl](#)

Formulareingaben

- [CGI und Perl](#)

Gästebücher

- [CGI und Perl](#)

HTTP-Header

- [CGI und Perl](#)

Kommunikationsmodell

- [CGI und Perl](#)

Location-Header

- [CGI und Perl](#)

Programme per SSI aufrufen

- [Erste serverseitige Techniken](#)

QUERY_STRING

- [CGI und Perl](#)

Serverseitige ImageMaps

- [CGI und Perl](#)

Sonderzeichen

- [CGI und Perl](#)

testen

- [CGI und Perl](#)

Umgebungsvariablen

- [CGI und Perl](#)

URL-Codierung

- [CGI und Perl](#)

Webseiten dynamisch erzeugen

- [CGI und Perl](#)

Webserver-Konfiguration

- [CGI und Perl](#)

Client-Side Pull

- [Fortgeschrittenes und dynamisches HTML](#)

Clipping

- [Positionieren mit Stylesheets](#)

Codierstil

- [Am Anfang war... HTML](#)

Common Gateway Interface

- [CGI und Perl](#)

Compaq

- [XML und XHTML](#)

Compiler

- [Wozu JavaScript?](#)

Cookies

- [Cookies](#)

-

[PHP - Einführung](#)

abfragen

- [Cookies](#)

-

[Active Server Pages](#)

-

[PHP - Einführung](#)

Aufbau

- [Cookies](#)

Browser

- [Cookies](#)

CGI

- [Cookies](#)

Einschränkungen

-

[Active Server Pages](#)

Gültigkeitsdatum

- [Cookies](#)
- [Active Server Pages](#)
- [PHP - Einführung](#)

Leer- und Sonderzeichen

- [Cookies](#)

löschen

- [Cookies](#)
- [Active Server Pages](#)
- [PHP - Einführung](#)

Pfadangaben

- [Cookies](#)

setzen

- [Cookies](#)
- [Active Server Pages](#)
- [PHP - Einführung](#)
- [PHP - Einführung](#)

zugehörige Webseiten

- [Cookies](#)

Copyright

- [Ein gelungener Webauftritt](#)

D

Date (JavaScript-Klasse)

- [Wozu JavaScript?](#)
- [JavaScript- Grundkurs](#)

Datenbanken

- [Datenbankanbindung](#)

Datenintegrität

- [Datenbankanbindung](#)

Datentypen

- [Datenbankanbindung](#)

Datumswerte

- [Datenbankanbindung](#)

Fremdschlüssel

- [Datenbankanbindung](#)

mit ASP

- [Datenbankanbindung](#)

mit Perl

- [Datenbankanbindung](#)

mit PHP

- [Datenbankanbindung](#)

Modellierung

- [Datenbankanbindung](#)

Online-Shop

- [Datenbankanbindung](#)

Primärschlüssel

- [Datenbankanbindung](#)

relational

- [Datenbankanbindung](#)

Spalten

- [Datenbankanbindung](#)

SQL

- [Datenbankanbindung](#)

Tabellen

- [Datenbankanbindung](#)

Datentypen

- [JavaScript- Grundkurs](#)

Datum

- [Wozu JavaScript?](#)
- [JavaScript- Grundkurs](#)
- [Erste serverseitige Techniken](#)

Dekrement

- [JavaScript- Grundkurs](#)

deprecated

- [Am Anfang war... HTML](#)
- [Am Anfang war... HTML](#)

Design

Animationen

- [Animationen](#)

Antialiasing

- [Am Anfang war... HTML](#)

Aspekte

- [Von der Idee zum eigenen Web](#)

Benutzeroberflächen

- [JavaScript und Formulare](#)

Bilder fürs Web aufbereiten

- [Am Anfang war... HTML](#)

Blinde GIFs

- [Formatieren mit Stylesheets](#)
- [Positionieren mit Stylesheets](#)

CSS-Positionierung

- [Positionieren mit Stylesheets](#)

Elemente überlagern

- [Positionieren mit Stylesheets](#)

Farben

- [Am Anfang war... HTML](#)

Frames

- [Tabellen, Frames und Formulare](#)

Hintergrundbilder

- [Am Anfang war... HTML](#)

ImageMaps

- [Am Anfang war... HTML](#)

Informationen für Suchmaschinen

- [Fortgeschrittenes und dynamisches HTML](#)

Inhaltsverzeichnisse

- [Am Anfang war... HTML](#)

modularer Seitenaufbau mit SSI

- [Erste serverseitige Techniken](#)

Navigationsleisten

- [Am Anfang war... HTML](#)

Schriften

- [Am Anfang war... HTML](#)

Seiten automatisch aktualisieren

- [Fortgeschrittenes und dynamisches HTML](#)

Seiten automatisch wechseln

- [Fortgeschrittenes und dynamisches HTML](#)

Seitenübergänge

- [Fortgeschrittenes und dynamisches HTML](#)

Tabellen

- [Tabellen, Frames und Formulare](#)

Tabellen versus CSS-Positionierung

- [Positionieren mit Stylesheets](#)

Todsünden

- [Ein gelungener Webauftritt](#)

Usability

- [Ein gelungener Webauftritt](#)

Zielsetzung

- [Von der Idee zum eigenen Web](#)

DHTML

Animationen

- [Animationen](#)

Browserunabhängige Lösungen

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Datum ausgeben

- [Wozu JavaScript?](#)

Definition

- [Wozu JavaScript?](#)

document-Objekt

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

dynamische Listen

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

dynamischen Text ausgeben

- [Wozu JavaScript?](#)

Hintergrundfarbe wechseln

- [Wozu JavaScript?](#)

HTML-Attribute verändern

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

HTML-Elemente über DOM-Knoten ansteuern

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

HTML-Elemente über ID ansteuern

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Rollover-Effekte für Schaltflächen

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Stileigenschaften verändern

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Text von Textelemente verändern

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Zugriff auf den Inhalt der Webseite

- [Mit JavaScript auf HTML-Elemente zugreifen](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

document (JavaScript-Objekt)

- [JavaScript- Grundkurs](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)
 - Array-Eigenschaften
 - [Mit JavaScript auf HTML-Elemente zugreifen](#)
 - getElementById()-Methode
 - [Mit JavaScript auf HTML-Elemente zugreifen](#)
 - getElementsByName()-Methode
 - [Mit JavaScript auf HTML-Elemente zugreifen](#)
 - Zugriff auf den Inhalt der Webseite
 - [Mit JavaScript auf HTML-Elemente zugreifen](#)
 - Zugriff auf Eigenschaften der Webseite
 - [Mit JavaScript auf HTML-Elemente zugreifen](#)

document.all

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

document.bgColor (JavaScript-Eigenschaft)

- [Wozu JavaScript?](#)

document.cookie (JavaScript-Objekt)

- [Cookies](#)

document.getElementById (JavaScript-Methode)

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

document.getElementsByName (JavaScript-Methode)

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

document.write (JavaScript-Methode)

- [Wozu JavaScript?](#)

Dokumentenverzeichnis (Server)

- [Von der Idee zum eigenen Web](#)

DOM-Modell

- [Mit JavaScript auf HTML-Elemente zugreifen](#)
 - Browserunterschiede
 - [Mit JavaScript auf HTML-Elemente zugreifen](#)
 - Dokumenthierarchie
 - [Mit JavaScript auf HTML-Elemente zugreifen](#)
 - Knoten ansteuern
 - [Mit JavaScript auf HTML-Elemente zugreifen](#)
 - Knoten bearbeiten
 - [Mit JavaScript auf HTML-Elemente zugreifen](#)
 - Navigieren der Dokumenthierarchie
 - [Mit JavaScript auf HTML-Elemente zugreifen](#)

DreamWeaver

- [Von der Idee zum eigenen Web](#)

- [Von der Idee zum eigenen Web](#)

DTD

- [XML und XHTML](#)

E

ECMA-Standard

- [Wozu JavaScript?](#)

Ereignisse

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

addEventListener()

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Argumente an Ereignisfunktionen übergeben

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

auf auslösendes HTML-Element zugreifen

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Auswahlfelder

- [JavaScript und Formulare](#)

Browserunterstützung

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Bubbling-Up

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

captureEvents()

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

event-Objekt

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

für Steuerelemente

- [JavaScript und Formulare](#)

globale Ereignisbearbeitung

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Informationen zum Ereignis

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Kontrollkästchen

- [JavaScript und Formulare](#)

Koordinaten eines Mausklicks

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

mit JavaScript-Code verbinden

- [Wozu JavaScript?](#)

onchange

- [JavaScript und Formulare](#)

onclick in Bildern

- [Wozu JavaScript?](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

onclick in Steuerelementen

- [Wozu JavaScript?](#)
- [JavaScript und Formulare](#)
- [JavaScript und Formulare](#)

onclick in Textabsätzen

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

onload

- [Wozu JavaScript?](#)

onmouseout

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

onmouseover

- [JavaScript- Grundkurs](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

onreset

- [JavaScript und Formulare](#)

onsubmit

- [JavaScript und Formulare](#)
- [JavaScript und Formulare](#)

onunload

- [Cookies](#)

Optionsfelder

- [JavaScript und Formulare](#)

Schaltflächen

- [JavaScript und Formulare](#)

simulieren

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Textfelder

- [JavaScript und Formulare](#)

this

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Übersicht

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Expat

- [XML und XHTML](#)

Farben

- [Am Anfang war... HTML](#)
- [Formatieren mit Stylesheets](#)
 - CSS-Schlüsselwörter
 - [Am Anfang war... HTML](#)
 - RGB-Kodierung
 - [Am Anfang war... HTML](#)
 - transparente Farbe
 - [Am Anfang war... HTML](#)

Flash-Animationen

- [Animationen](#)
 - Ablaufgeschwindigkeit
 - [Animationen](#)
 - Animation testen
 - [Animationen](#)
 - Arbeitsbereich
 - [Animationen](#)
 - Bilder importieren
 - [Animationen](#)
 - Bildrate
 - [Animationen](#)
 - Ebenen
 - [Animationen](#)
 - Flash-Player
 - [Animationen](#)
 - Frames kopieren
 - [Animationen](#)
 - Frames zeichnen
 - [Animationen](#)
 - Größe der Leinwand
 - [Animationen](#)
 - Hilfsgitter
 - [Animationen](#)
 - Hintergründe
 - [Animationen](#)
 - in Webseiten einbinden
 - [Animationen](#)
 - Laufpfade (Motion Tween)
 - [Animationen](#)
 - Morphing
 - [Animationen](#)

Motion Tween

- [Animationen](#)

neue Animation beginnen

- [Animationen](#)

Objekte verschieben

- [Animationen](#)

Schlüsselframes

- [Animationen](#)

Schreibschutz (für Ebenen)

- [Animationen](#)

Shape Tween

- [Animationen](#)

statische Inhalte

- [Animationen](#)

Zeichenwerkzeuge

- [Animationen](#)

Zeitleiste

- [Animationen](#)

Zwiebelhautdarstellung

- [Animationen](#)

Zwischenbilder

- [Animationen](#)

Formatierung

alle Vorkommen eines HTML-Tags formatieren

- [Formatieren mit Stylesheets](#)

einzelne HTML-Elemente formatieren

- [Formatieren mit Stylesheets](#)

Hintergrund von Tabellenzellen

- [Formatieren mit Stylesheets](#)

Hyperlinks

- [Formatieren mit Stylesheets](#)

Inline-Stile

- [Am Anfang war... HTML](#)

Listen

- [Am Anfang war... HTML](#)
- [Formatieren mit Stylesheets](#)

logische HTML-Formatierung

- [Am Anfang war... HTML](#)

physikalische HTML-Formatierung

- [Am Anfang war... HTML](#)

Seitenabschnitte

- [Am Anfang war... HTML](#)

selbst definierte Gruppen von HTML-Elementen formatieren

- [Formatieren mit Stylesheets](#)

Stileigenschaften

- [Formatieren mit Stylesheets](#)

Stylesheets

- [Formatieren mit Stylesheets](#)

Tabellen

- [Tabellen, Frames und Formulare](#)
- [Formatieren mit Stylesheets](#)

Textpassagen

- [Am Anfang war... HTML](#)

traditionelle

- [Am Anfang war... HTML](#)

Formulare

- [Tabellen, Frames und Formulare](#)
- [Tabellen, Frames und Formulare](#)

Benutzeroberflächen

- [JavaScript und Formulare](#)

CGI-Verarbeitung

- [CGI und Perl](#)

Eingaben verarbeiten

- [JavaScript und Formulare](#)

Formularfelder

- [Tabellen, Frames und Formulare](#)

Gästebücher

- [CGI und Perl](#)

JavaScript

- [JavaScript und Formulare](#)

Perl-CGI-Programme als Bearbeiter angeben

- [CGI und Perl](#)

Radiobuttons

- [Active Server Pages](#)

verborgene Felder

- [CGI und Perl](#)

Verifizierung von Formulareingaben

- [JavaScript und Formulare](#)

Vollständigkeitsprüfung

- [JavaScript und Formulare](#)

Formularfelder

- [Tabellen, Frames und Formulare](#)

for-Schleife

- [JavaScript- Grundkurs](#)

Frames

- [Tabellen, Frames und Formulare](#)

alternative Webseiten

- [Tabellen, Frames und Formulare](#)

Bildlaufleisten

- [Tabellen, Frames und Formulare](#)

definieren

- [Tabellen, Frames und Formulare](#)

DOCTYPE-Versionsinformation

- [Tabellen, Frames und Formulare](#)

Frameseite

- [Tabellen, Frames und Formulare](#)

Frameseite über Hyperlink verlassen

- [JavaScript und Frames](#)

Hyperlinks

- [Tabellen, Frames und Formulare](#)

Inline-Frames

- [Tabellen, Frames und Formulare](#)

Innenabstand

- [Tabellen, Frames und Formulare](#)

JavaScript

- [JavaScript und Frames](#)

JavaScript-unterstützte Hyperlinks

- [JavaScript und Frames](#)

Javascript-Zugriff auf HTML-Elemente

- [JavaScript und Frames](#)

konfigurieren

- [Tabellen, Frames und Formulare](#)

mehrere Frames gleichzeitig laden

- [JavaScript und Frames](#)

Rahmen ausblenden

- [Tabellen, Frames und Formulare](#)

typische Frame-Designs

- [Tabellen, Frames und Formulare](#)

verschachteln

- [Tabellen, Frames und Formulare](#)

verschiebbare Rahmen

- [Tabellen, Frames und Formulare](#)

FrontPage

- [Von der Idee zum eigenen Web](#)
- [Von der Idee zum eigenen Web](#)

FTP

- [Von der Idee zum eigenen Web](#)

FTS_PRO

- [Von der Idee zum eigenen Web](#)

Funktionen

- [JavaScript- Grundkurs](#)

Argumente

- [JavaScript- Grundkurs](#)

Aufruf

- [JavaScript- Grundkurs](#)

Call by value

- [JavaScript- Grundkurs](#)

Definition

- [JavaScript- Grundkurs](#)

Parameter

- [JavaScript- Grundkurs](#)

Rückgabewerte

- [JavaScript- Grundkurs](#)

G

Gästebücher

- [CGI und Perl](#)

mit ASP

- [Datenbankanbindung](#)

mit PHP

- [Datenbankanbindung](#)

GIF

- [Am Anfang war... HTML](#)

Blinde GIFs

- [Formatieren mit Stylesheets](#)
- [Positionieren mit Stylesheets](#)

GIF-Animationen

- [Animationen](#)

Groß- und Kleinschreibung

HTML

- [Am Anfang war... HTML](#)

Java

- [Java-Applets](#)

JavaScript

- [Wozu JavaScript?](#)

Perl

- [CGI und Perl](#)

PHP

- [PHP - Einführung](#)

VBScript

- [Active Server Pages](#)

XHTML

- [XML und XHTML](#)

Gültigkeitsbereiche

- [JavaScript- Grundkurs](#)

H

Header-Abschnitt

- [Am Anfang war... HTML](#)

Hintergrundbilder

- [Am Anfang war... HTML](#)
- [Am Anfang war... HTML](#)
- [Formatieren mit Stylesheets](#)

history (JavaScript-Objekt)

- [JavaScript- Grundkurs](#)

HTML

- [Tabellen, Frames und Formulare](#)

Absätze

- [Am Anfang war... HTML](#)

Attribute

- [Am Anfang war... HTML](#)

Aufzählungen

- [Am Anfang war... HTML](#)

Basisadresse

- [Fortgeschrittenes und dynamisches HTML](#)

Bilder

- [Am Anfang war... HTML](#)

Block-Elemente

- [Formatieren mit Stylesheets](#)

Body-Abschnitt

- [Am Anfang war... HTML](#)

Boolesche Attribute

- [Am Anfang war... HTML](#)
- [XML und XHTML](#)

Codierstil

- [Am Anfang war... HTML](#)

CSS-Positionierung

- [Positionieren mit Stylesheets](#)

Definitionen

- [Am Anfang war... HTML](#)

deprecated

- [Am Anfang war... HTML](#)

DOM-Modell

- [Mit JavaScript auf HTML-Elemente zugreifen](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Elemente

- [Am Anfang war... HTML](#)
-

Farben

- [Am Anfang war... HTML](#)

Formulare

- [Tabellen, Frames und Formulare](#)

Geschichte

- [Am Anfang war... HTML](#)

Groß- und Kleinschreibung

- [Am Anfang war... HTML](#)

Grundgerüst

- [Am Anfang war... HTML](#)

Header-Abschnitt

- [Am Anfang war... HTML](#)

Hintergrund eines Elements

- [Formatieren mit Stylesheets](#)

Hintergrundbilder

- [Am Anfang war... HTML](#)

horizontale Linien

- [Am Anfang war... HTML](#)

Hyperlinks

- [Am Anfang war... HTML](#)

Inline-Elemente

- [Formatieren mit Stylesheets](#)

Kommentare

- [Am Anfang war... HTML](#)

Kurzinformationen

- [Fortgeschrittenes und dynamisches HTML](#)

Listen

- [Am Anfang war... HTML](#)

Logische Formatierungen

- [Am Anfang war... HTML](#)

Markups

- [Am Anfang war... HTML](#)

Metainformationen

- [Fortgeschrittenes und dynamisches HTML](#)

Multimedia

- [Fortgeschrittenes und dynamisches HTML](#)

Nummerierungen

- [Am Anfang war... HTML](#)

Parameter an Applets übergeben

- [Java-Applets](#)

Physikalische Formatierungen

- [Am Anfang war... HTML](#)

Plattformunabhängigkeit

- [Am Anfang war... HTML](#)

Quickinfos

- [Fortgeschrittenes und dynamisches HTML](#)

Ränder

- [Formatieren mit Stylesheets](#)

Rahmen

- [Formatieren mit Stylesheets](#)

Schriften

- [Am Anfang war... HTML](#)

Server Side Includes

- [Erste serverseitige Techniken](#)

Sichtbarkeit

- [Positionieren mit Stylesheets](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Sonderzeichen

- [Am Anfang war... HTML](#)

- Sounddateien
 - [Fortgeschrittenes und dynamisches HTML](#)
- Soundformate
 - [Fortgeschrittenes und dynamisches HTML](#)
- Sprachcodes
 - [Fortgeschrittenes und dynamisches HTML](#)
- Standard-Skriptsprache angeben
 - [Wozu JavaScript?](#)
- Steuerelemente
 - [Tabellen, Frames und Formulare](#)
- Tabellen
 - [Tabellen, Frames und Formulare](#)
- Tags
 - [Am Anfang war... HTML](#)
 - [Am Anfang war... HTML](#)
- Text
 - [Am Anfang war... HTML](#)
 - [Formatieren mit Stylesheets](#)
- Textpassagen auszeichnen
 - [Am Anfang war... HTML](#)
- Titel der Webseite
 - [Am Anfang war... HTML](#)
- traditionelle Positionierung
 - [Positionieren mit Stylesheets](#)
 - [Positionieren mit Stylesheets](#)
- Überschriften
 - [Am Anfang war... HTML](#)
- URLs
 - [Fortgeschrittenes und dynamisches HTML](#)
- Versionsinformation
 - [Am Anfang war... HTML](#)
- Videos
 - [Fortgeschrittenes und dynamisches HTML](#)
- vorformatierter Text
 - [Am Anfang war... HTML](#)
- Zeilenumbrüche
 - [Am Anfang war... HTML](#)
- zu XHTML
 - [XML und XHTML](#)

HTML-Elemente

- [Am Anfang war... HTML](#)

HTML-Kit

- [Von der Idee zum eigenen Web](#)

HTTP

-

[Active Server Pages](#)

Antwortheader

- [Fortgeschrittenes und dynamisches HTML](#)

Content-type-Header

- [CGI und Perl](#)

Header

- [CGI und Perl](#)

Header simulieren

- [Fortgeschrittenes und dynamisches HTML](#)

Location-Header

- [CGI und Perl](#)

Hyperlinks

- [Am Anfang war... HTML](#)

Bilder

- [Am Anfang war... HTML](#)

Formatierung

- [Formatieren mit Stylesheets](#)

Frames

- [Tabellen, Frames und Formulare](#)

Frameseite verlassen

- [JavaScript und Frames](#)

ImageMaps

- [Am Anfang war... HTML](#)

in neues Fenster laden

- [Tabellen, Frames und Formulare](#)

JavaScript-unterstützte Frame-Hyperlinks

- [JavaScript und Frames](#)

mehrere Frames gleichzeitig laden

- [JavaScript und Frames](#)

mit JavaScript-Code verbinden

- [Wozu JavaScript?](#)

relative

- [Am Anfang war... HTML](#)

URL-Basis

- [Am Anfang war... HTML](#)

Webseiten in neues Browserfenster laden

- [Am Anfang war... HTML](#)

Zeichensatz

- [Am Anfang war... HTML](#)

Zielframes

- [Tabellen, Frames und Formulare](#)

zu anderen Protokolle

- [Am Anfang war... HTML](#)

zu anderen Textstellen

- [Am Anfang war... HTML](#)

zu anderen Webseiten der eigenen Site

- [Am Anfang war... HTML](#)

zu anderen Websites

- [Am Anfang war... HTML](#)

if-Verzweigung

- [JavaScript- Grundkurs](#)

ImageMaps

- [Am Anfang war... HTML](#)

clientseitige

- [Am Anfang war... HTML](#)

serverseitige

- [Am Anfang war... HTML](#)

- [CGI und Perl](#)

Inhaltsverzeichnisse

- [Am Anfang war... HTML](#)
- [Am Anfang war... HTML](#)
- [Tabellen, Frames und Formulare](#)

Inkrement

- [JavaScript- Grundkurs](#)

Inline-Frames

- [Tabellen, Frames und Formulare](#)

innerHTML

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Interlacing

- [Am Anfang war... HTML](#)

Internet Service Provider

- [Von der Idee zum eigenen Web](#)

Interpreter

- [Wozu JavaScript?](#)

IP-Adresse

- [Von der Idee zum eigenen Web](#)

J

Java

.class-Datei

- [Java-Applets](#)
- [Java-Applets](#)

Animationen

- [Animationen](#)

Applets

- [Java-Applets](#)

Appletviewer

- [Java-Applets](#)

Bibliotheken

- [Java-Applets](#)

Browser

- [Java-Applets](#)

Bytecode

- [Java-Applets](#)

class (Schlüsselwort)

- [Java-Applets](#)

Compiler

- [Java-Applets](#)
- [Java-Applets](#)

Datentypen

- [Java-Applets](#)

Entwicklungsumgebung

- [Java-Applets](#)

Ereignismethoden

- [Java-Applets](#)

Ereignisse

- [Java-Applets](#)

Exception-Behandlung

- [Animationen](#)

extends (Schlüsselwort)

- [Java-Applets](#)

Farben

- [Java-Applets](#)
- [Java-Applets](#)

Groß- und Kleinschreibung

- [Java-Applets](#)

import (Schlüsselwort)

- [Java-Applets](#)

Instanzvariablen

- [Java-Applets](#)

Interpreter

- [Java-Applets](#)

javac

- [Java-Applets](#)

Klassen

- [Java-Applets](#)

Kommentare

- [Java-Applets](#)

Konstruktoren

- [Java-Applets](#)

Methoden überschreiben

- [Java-Applets](#)

Packages

- [Java-Applets](#)

Plattformunabhängigkeit

- [Java-Applets](#)

public (Schlüsselwort)

- [Java-Applets](#)

Quelldatei

- [Java-Applets](#)
- [Java-Applets](#)

Schnittstellen

- [Animationen](#)

Schriftcharakteristika abfragen

- [Animationen](#)

Sicherheit

- [Java-Applets](#)

Swing

- [Java-Applets](#)

Threads

- [Animationen](#)

Typenumwandlung

- [Java-Applets](#)

Variablen

- [Java-Applets](#)
- [Java-Applets](#)

Vererbung

- [Java-Applets](#)

Zeichenmethoden

- [Java-Applets](#)

JavaScript

Abbruchbefehle für Schleifen

- [JavaScript- Grundkurs](#)

alternativer HTML-Code

- [Wozu JavaScript?](#)

Animationen

- [Animationen](#)

Anweisungen

- [JavaScript- Grundkurs](#)

Arrays

- [JavaScript- Grundkurs](#)

Ausdrücke

- [JavaScript- Grundkurs](#)

Ausführung

- [Wozu JavaScript?](#)
- [Wozu JavaScript?](#)

Auswahlfelder

- [JavaScript und Formulare](#)

Benutzereingaben

- [JavaScript- Grundkurs](#)

Browser

- [Wozu JavaScript?](#)

Browser-Objekte

- [JavaScript- Grundkurs](#)

Browserunabhängige Lösungen

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Code verbergen

- [Wozu JavaScript?](#)

Cookies

- [Cookies](#)

Datentypen

- [JavaScript- Grundkurs](#)
- [JavaScript- Grundkurs](#)

Datum

- [Wozu JavaScript?](#)
- [JavaScript- Grundkurs](#)

debuggen

- [Wozu JavaScript?](#)

Dekrement

- [JavaScript- Grundkurs](#)

document.write

- [Wozu JavaScript?](#)

document-Objekt

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

DOM-Modell

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

do-while-Schleife

- [JavaScript- Grundkurs](#)

dynamischen Text ausgeben

- [Wozu JavaScript?](#)

Einbindung in Webseiten

- [Wozu JavaScript?](#)

EMCA-Standard

- [Wozu JavaScript?](#)

Ereignisse

- [Wozu JavaScript?](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

externe Skripte

- [Wozu JavaScript?](#)

Fehlersuche in Code

- [Wozu JavaScript?](#)

for-in-Schleife

- [JavaScript- Grundkurs](#)

Formulare

- [JavaScript und Formulare](#)

for-Schleife

- [JavaScript- Grundkurs](#)

Frame-Hyperlinks

- [JavaScript und Frames](#)

Frames

- [JavaScript und Frames](#)

Frameseite über Hyperlink verlassen

- [JavaScript und Frames](#)

Funktionen

- [Wozu JavaScript?](#)
- [JavaScript- Grundkurs](#)

Geschichte

- [Wozu JavaScript?](#)

Groß- und Kleinschreibung

- [Wozu JavaScript?](#)

Gültigkeitsbereiche

- [JavaScript- Grundkurs](#)

HTML-Attribute verändern

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

HTML-Elemente über DOM-Knoten ansteuern

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

HTML-Elemente über ID ansteuern

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

if-Verzweigung

- [JavaScript- Grundkurs](#)

im <body>-Abschnitt

- [Wozu JavaScript?](#)

implizite Variablendefinition

- [JavaScript- Grundkurs](#)

Inkrement

- [JavaScript- Grundkurs](#)

Klammern

- [JavaScript- Grundkurs](#)

Klassen

- [JavaScript- Grundkurs](#)

Kommentare

- [Wozu JavaScript?](#)

Konstanten

- [JavaScript- Grundkurs](#)

Kontrollkästchen

- [JavaScript und Formulare](#)

Kontrollstrukturen

- [JavaScript- Grundkurs](#)

mathematische Funktionen

- [JavaScript- Grundkurs](#)

mehrere Frames gleichzeitig laden

- [JavaScript und Frames](#)

Meldungsfenster

- [Wozu JavaScript?](#)

mit Hyperlink verbinden

- [Wozu JavaScript?](#)

Modulo

- [JavaScript- Grundkurs](#)

Namensgebung

- [JavaScript- Grundkurs](#)

neue Webseite laden

- [JavaScript- Grundkurs](#)

neues Browserfenster öffnen

- [Wozu JavaScript?](#)

Objekte

- [JavaScript- Grundkurs](#)

Operatoren

- [JavaScript- Grundkurs](#)

Optionsfelder

- [JavaScript und Formulare](#)

periodischer Funktionsaufruf

- [Animationen](#)

Schaltflächen

- [JavaScript und Formulare](#)

Schleifen

- [JavaScript- Grundkurs](#)

Schlüsselwörter

- [JavaScript- Grundkurs](#)

serverseitiges Skripting

- [Erste serverseitige Techniken](#)

Sicherheitsbedenken

- [Wozu JavaScript?](#)

Standard-Skriptsprache angeben

- [Wozu JavaScript?](#)

Statusleiste des Browsers

- [Wozu JavaScript?](#)

Stileigenschaften verändern

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Strings

- [JavaScript- Grundkurs](#)
- [JavaScript- Grundkurs](#)

switch-Verzweigung

- [JavaScript- Grundkurs](#)

Syntax

- [JavaScript- Grundkurs](#)

Text von Textelemente verändern

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Textfelder

- [JavaScript und Formulare](#)

Typenumwandlung

- [JavaScript- Grundkurs](#)

undefined

- [JavaScript- Grundkurs](#)

Variablen

- [JavaScript- Grundkurs](#)

Verdeckung

- [JavaScript- Grundkurs](#)

Vergleich zu Java

- [Wozu JavaScript?](#)

Vergleiche

- [JavaScript- Grundkurs](#)

vordefinierte Klassen

- [JavaScript- Grundkurs](#)

Vordergrundfarbe

- [JavaScript- Grundkurs](#)

Webseite neu laden lassen

- [JavaScript- Grundkurs](#)

while-Schleife

- [JavaScript- Grundkurs](#)

Zufallszahlen

- [JavaScript- Grundkurs](#)

Zugriff auf das Browserfenster

- [JavaScript- Grundkurs](#)

Zugriff auf den Browser

- [JavaScript- Grundkurs](#)

Zugriff auf den Inhalt der Webseite

- [JavaScript- Grundkurs](#)

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Zugriff auf den URI der Webseite

- [JavaScript- Grundkurs](#)

Zugriff auf die History des Browsers

- [JavaScript- Grundkurs](#)

Zugriff auf Eigenschaften der Webseite

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Zugriff auf Formulare

- [JavaScript und Formulare](#)

Zugriff auf HTML-Elemente in Frames

- [JavaScript und Frames](#)

Zugriff auf Steuerelemente

- [JavaScript und Formulare](#)

javascript

- [Wozu JavaScript?](#)

JPG

- [Am Anfang war... HTML](#)

JScript

- [Wozu JavaScript?](#)

JS-Dateien

- [Wozu JavaScript?](#)

K

Klammern

- [JavaScript- Grundkurs](#)

Klassen

- [JavaScript- Grundkurs](#)

definieren

- [JavaScript- Grundkurs](#)

Eigenschaften

- [JavaScript- Grundkurs](#)

Methoden

- [JavaScript- Grundkurs](#)

Objekte erzeugen

- [JavaScript- Grundkurs](#)

this

- [JavaScript- Grundkurs](#)

Zugriff auf Elemente

- [JavaScript- Grundkurs](#)

Kommentare

HTML

- [Am Anfang war... HTML](#)

JavaScript

- [Wozu JavaScript?](#)

Perl

- [CGI und Perl](#)

PHP

- [PHP - Einführung](#)

Stylesheets

- [Formatieren mit Stylesheets](#)

VBScript

- [Active Server Pages](#)

Kompilation

- [Wozu JavaScript?](#)

Konsole

- [Java-Applets](#)

Konstanten

- [JavaScript- Grundkurs](#)

Kontrollstrukturen

- [JavaScript- Grundkurs](#)

Kurzinformationen

- [Fortgeschrittenes und dynamisches HTML](#)

L

L- und R-Wert

- [JavaScript- Grundkurs](#)

LAMP

- [PHP - Einführung](#)

Laufschriften

- [Animationen](#)

Leerzeichen

in HTML-Code

- [Am Anfang war... HTML](#)

in Tabellenzellen

- [Tabellen, Frames und Formulare](#)

neben Tags

- [Am Anfang war... HTML](#)

Linien

- [Am Anfang war... HTML](#)

Listen

- [Am Anfang war... HTML](#)

Aufzählungen

- [Am Anfang war... HTML](#)

Definitionen

- [Am Anfang war... HTML](#)

dynamische

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

formatieren

- [Am Anfang war... HTML](#)

Nummerierungen

- [Am Anfang war... HTML](#)

spezielle Stileigenschaften

- [Formatieren mit Stylesheets](#)

Symbole

- [Am Anfang war... HTML](#)

verschachteln

- [Am Anfang war... HTML](#)

Literale

- [JavaScript- Grundkurs](#)

Literaturhinweise

- [HTML](#)

LiveScript

- [Wozu JavaScript?](#)

localhost

- [Von der Idee zum eigenen Web](#)

location (JavaScript-Objekt)

- [JavaScript- Grundkurs](#)

Lösungen

- [Tag 1: Von der Idee zum eigenen Web](#)

M

Markups

- [Am Anfang war... HTML](#)

Maschinencode

- [Wozu JavaScript?](#)

Math (JavaScript-Klasse)

- [JavaScript- Grundkurs](#)

- [JavaScript- Grundkurs](#)

Math.random (JavaScript-Methode)

- [JavaScript- Grundkurs](#)

Math.sqrt (JavaScript-Methode)

- [JavaScript- Grundkurs](#)

Medien-Typen

- [Formatieren mit Stylesheets](#)

Meldungsfenster

- [Wozu JavaScript?](#)

Metainformationen

- [Fortgeschrittenes und dynamisches HTML](#)

Microsoft Word

- [Von der Idee zum eigenen Web](#)

Modulo

- [JavaScript- Grundkurs](#)

MSDOS-Eingabeaufforderung

- [Java-Applets](#)

Multimedia

- [Fortgeschrittenes und dynamisches HTML](#)

<embed>-Tag

- [Fortgeschrittenes und dynamisches HTML](#)

<object>-Tag

- [Fortgeschrittenes und dynamisches HTML](#)

Sounddateien

- [Fortgeschrittenes und dynamisches HTML](#)

Videos

- [Fortgeschrittenes und dynamisches HTML](#)

MySQL

- [PHP - Einführung](#)

N

Navigationsleisten

- [Am Anfang war... HTML](#)

navigator (JavaScript-Objekt)

- [JavaScript- Grundkurs](#)

new

- [JavaScript- Grundkurs](#)

Notepad-Editor

- [Von der Idee zum eigenen Web](#)

Nummerierungszeichen

- [Am Anfang war... HTML](#)

0

Objekte

- [JavaScript- Grundkurs](#)

ODBC

- [PHP - Einführung](#)
- [Datenbankanbindung](#)
- [Datenbankanbindung](#)
DSN
 - [Datenbankanbindung](#)

OmniHttpd

- [Von der Idee zum eigenen Web](#)
- [Der Apache-Server](#)

onchange (JavaScript-Ereignis)

- [JavaScript und Formulare](#)

onclick (JavaScript-Ereignis)

- [Wozu JavaScript?](#)
- [Wozu JavaScript?](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)
- [JavaScript und Formulare](#)
- [JavaScript und Formulare](#)

Online-Shop

- [Datenbankanbindung](#)

onload (JavaScript-Ereignis)

- [Wozu JavaScript?](#)

onmouseout (JavaScript-Ereignis)

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

onmouseover (JavaScript-Ereignis)

- [JavaScript- Grundkurs](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

onreset (JavaScript-Ereignis)

- [JavaScript und Formulare](#)

onsubmit (JavaScript-Ereignis)

- [JavaScript und Formulare](#)
- [JavaScript und Formulare](#)

onunload (JavaScript-Ereignis)

- [Cookies](#)

OOP (Objektorientierte Programmierung)

Klassen

- [JavaScript- Grundkurs](#)

open (JavaScript-Methode)

- [Wozu JavaScript?](#)

Operatoren

- [JavaScript- Grundkurs](#)

Optionsfelder

- [Tabellen, Frames und Formulare](#)

Ereignisbehandlung

- [JavaScript und Formulare](#)

gruppieren

- [JavaScript und Formulare](#)

P

Paint Shop Pro

- [Am Anfang war... HTML](#)

Parameter

- [JavaScript- Grundkurs](#)

parseFloat (JavaScript-Methode)

- [JavaScript- Grundkurs](#)

parseInt (JavaScript-Methode)

- [JavaScript- Grundkurs](#)
- [JavaScript und Formulare](#)

Perl

- [CGI und Perl](#)

(Kommentar)

- [CGI und Perl](#)

\$ (Skalare)

- [CGI und Perl](#)

% (Hashes)

- [CGI und Perl](#)

@ (Arrays)

- [CGI und Perl](#)

Arrays

- [CGI und Perl](#)

Ausgabe

- [CGI und Perl](#)

chomp

- [CGI und Perl](#)

CPAN

- [CGI und Perl](#)

Dateien

- [CGI und Perl](#)

Eingabe

- [CGI und Perl](#)

Escape-Zeichen

- [CGI und Perl](#)

foreach-Schleife

- [CGI und Perl](#)

for-Schleife

- [CGI und Perl](#)

Funktionen

- [CGI und Perl](#)

Groß- und Kleinschreibung

- [CGI und Perl](#)

Hashes

- [CGI und Perl](#)
- [CGI und Perl](#)

HERE-Texte

- [CGI und Perl](#)

if-Bedingung

- [CGI und Perl](#)

installieren

- [CGI und Perl](#)

Kommentare

- [CGI und Perl](#)

Konstanten

- [CGI und Perl](#)

Kontrollstrukturen

- [CGI und Perl](#)

Module

- [CGI und Perl](#)

my (Schlüsselwort)

- [CGI und Perl](#)

Operatoren

- [CGI und Perl](#)

Shebang-Kommentar

- [CGI und Perl](#)

Shebang-Zeile

- [CGI und Perl](#)

Sonderzeichen

- [CGI und Perl](#)
- [CGI und Perl](#)
- [CGI und Perl](#)
- [CGI und Perl](#)

Strings

- [CGI und Perl](#)

Strings in Arrays verwandeln

- [CGI und Perl](#)

Umgebungsvariablen

- [CGI und Perl](#)
- [CGI und Perl](#)

Variablen

- [CGI und Perl](#)

Webserver-Konfiguration

- [CGI und Perl](#)

while-Schleife

- [CGI und Perl](#)

Personal Web Server

- [Von der Idee zum eigenen Web](#)

PHP

- [PHP - Einführung](#)

(Kommentar)

- [PHP - Einführung](#)

\$HTTP_COOKIE_VARS

- [PHP - Einführung](#)

\$HTTP_GET_VARS

■

[PHP - Einführung](#)

\$HTTP_POST_FILES

- [PHP - Einführung](#)

\$HTTP_POST_VARS

- [PHP - Einführung](#)

\$PHP_SELF

- [PHP - Einführung](#)

.

- [PHP - Einführung](#)

/* */ (Kommentar)

- [PHP - Einführung](#)

// (Kommentar)

- [PHP - Einführung](#)

<?php ?>

- [PHP - Einführung](#)

array

- [PHP - Einführung](#)

Arrays

- [PHP - Einführung](#)

Cookies

- [PHP - Einführung](#)

count

- [PHP - Einführung](#)

Dateien

- [PHP - Einführung](#)

Datei-Upload

- [PHP - Einführung](#)

Datenbankzugriff

- [Datenbankanbindung](#)

Datumswerte

- [PHP - Einführung](#)

else-Anweisung

- [PHP - Einführung](#)

elseif-Anweisung

- [PHP - Einführung](#)

fopen

- [PHP - Einführung](#)

Formulare

- [PHP - Einführung](#)

Gästebuch

- [Datenbankanbindung](#)

Geschichte

- [PHP - Einführung](#)

getdate

- [PHP - Einführung](#)

- [PHP - Einführung](#)

getenv

- [PHP - Einführung](#)

Groß- und Kleinschreibung

- [PHP - Einführung](#)

header

- [PHP - Einführung](#)

Hintergrundfarbe speichern

- [PHP - Einführung](#)

if-Anweisung

- [PHP - Einführung](#)

Interpolation

- [PHP - Einführung](#)

Kommentare

- [PHP - Einführung](#)

Konkatenation

- [PHP - Einführung](#)

Kurzform

- [PHP - Einführung](#)

MySQL

- [PHP - Einführung](#)
- [Datenbankanbindung](#)

mysql_fetch_array

- [Datenbankanbindung](#)

mysql_query

- [Datenbankanbindung](#)

Namensgebung

- [PHP - Einführung](#)

ODBC

- [PHP - Einführung](#)
- [Datenbankanbindung](#)

odbc_connect

- [Datenbankanbindung](#)

odbc_exec

- [Datenbankanbindung](#)

odbc_fetch_row

- [Datenbankanbindung](#)

odbc_result

- [Datenbankanbindung](#)

Operatoren

PHP Documentation Group

- [PHP - Einführung](#)

php.ini

- [PHP - Einführung](#)

- [PHP - Einführung](#)

phpinfo

- [PHP - Einführung](#)

print

- [PHP - Einführung](#)

- [PHP - Einführung](#)

Schleifen

setcookie

- [PHP - Einführung](#)

str_replace

- [PHP - Einführung](#)
- [Datenbankanbindung](#)

strlen

- [PHP - Einführung](#)

strpos

- [PHP - Einführung](#)

strrpos

- [PHP - Einführung](#)

substr

- [PHP - Einführung](#)

switch-Anweisung

- [PHP - Einführung](#)

Tagesdatum

- [PHP - Einführung](#)

Text ausgeben

- [PHP - Einführung](#)

- [PHP - Einführung](#)

Umleitung

-

[PHP - Einführung](#)

Variablentypen

- [PHP - Einführung](#)

Version 3 und 4

- [PHP - Einführung](#)

Webserver

- [PHP - Einführung](#)

Wochentag

- [PHP - Einführung](#)

Zeichenketten

- [PHP - Einführung](#)

Zuweisung

- [PHP - Einführung](#)

PNG

- [Am Anfang war... HTML](#)

Positionierung

- [Positionieren mit Stylesheets](#)

absolute

- [Positionieren mit Stylesheets](#)

Attribute

- [Positionieren mit Stylesheets](#)

Bilder

- [Positionieren mit Stylesheets](#)
- [Positionieren mit Stylesheets](#)

Clipping

- [Positionieren mit Stylesheets](#)

Elemente überlagern

- [Positionieren mit Stylesheets](#)

fixierte

- [Positionieren mit Stylesheets](#)

Kontext

- [Positionieren mit Stylesheets](#)

position-Eigenschaft

- [Positionieren mit Stylesheets](#)

relative

- [Positionieren mit Stylesheets](#)

Sichtbarkeit

- [Positionieren mit Stylesheets](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

statische

- [Positionieren mit Stylesheets](#)

traditionelle

- [Positionieren mit Stylesheets](#)

versus Tabellen

- [Positionieren mit Stylesheets](#)

z-index

- [Positionieren mit Stylesheets](#)

Programmierung

Anweisungen

- [JavaScript- Grundkurs](#)

Arrays

- [JavaScript- Grundkurs](#)

Ausdrücke

- [JavaScript- Grundkurs](#)

Benutzereingaben

- [JavaScript- Grundkurs](#)

Binärcodierung

- [JavaScript- Grundkurs](#)

clientseitig

- [Wozu JavaScript?](#)

Compiler

- [Wozu JavaScript?](#)

Datentypen

- [JavaScript- Grundkurs](#)

Dekrement

- [JavaScript- Grundkurs](#)

for-Schleife

- [JavaScript- Grundkurs](#)

Funktionen

- [JavaScript- Grundkurs](#)

Gültigkeitsbereiche

- [JavaScript- Grundkurs](#)

if-Verzweigung

- [JavaScript- Grundkurs](#)

Inkrement

- [JavaScript- Grundkurs](#)

Interpreter

- [Wozu JavaScript?](#)

Klammern

- [JavaScript- Grundkurs](#)

Klassen

- [JavaScript- Grundkurs](#)

Kompilation

- [Wozu JavaScript?](#)

Konstanten

- [JavaScript- Grundkurs](#)

Kontrollstrukturen

- [JavaScript- Grundkurs](#)

L- und R-Wert

- [JavaScript- Grundkurs](#)

Maschinencode

- [Wozu JavaScript?](#)

Modulo

- [JavaScript- Grundkurs](#)

Objekte

- [JavaScript- Grundkurs](#)

Operatoren

- [JavaScript- Grundkurs](#)

Quelltext

- [Wozu JavaScript?](#)

Schleifen

- [JavaScript- Grundkurs](#)

serverseitige

- [Wozu JavaScript?](#)

Strings

- [JavaScript- Grundkurs](#)

switch-Verzweigung

- [JavaScript- Grundkurs](#)

Typenumwandlung

- [JavaScript- Grundkurs](#)

Variablen

- [JavaScript- Grundkurs](#)

Verdeckung

- [JavaScript- Grundkurs](#)

Vergleiche

- [JavaScript- Grundkurs](#)

while-Schleife

- [JavaScript- Grundkurs](#)

Zufallszahlen

- [JavaScript- Grundkurs](#)

prompt (JavaScript-Methode)

- [JavaScript- Grundkurs](#)

Protokolle

- [Am Anfang war... HTML](#)

Q

Quelltext

- [Wozu JavaScript?](#)

Quickinfos

- [Fortgeschrittenes und dynamisches HTML](#)

R

Rahmen

- [Formatieren mit Stylesheets](#)

Rand

- [Formatieren mit Stylesheets](#)

releaseEvents (proprietäre JavaScript-Methode)

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Rollover-Effekte

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Rückgabewerte

- [JavaScript- Grundkurs](#)

S

Schaltflächen

- [Tabellen, Frames und Formulare](#)

Abschicken (Submit)

- [Tabellen, Frames und Formulare](#)
- [Tabellen, Frames und Formulare](#)
- [JavaScript und Formulare](#)
- [JavaScript und Formulare](#)

Rollover-Effekte

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Zurücksetzen (Reset)

- [Tabellen, Frames und Formulare](#)
- [Tabellen, Frames und Formulare](#)
- [JavaScript und Formulare](#)

Schaltflächen (grafische)

- [Am Anfang war... HTML](#)

Schleifen

- [JavaScript- Grundkurs](#)

Schriften

- [Am Anfang war... HTML](#)
- [Formatieren mit Stylesheets](#)

Seitenübergänge

- [Fortgeschrittenes und dynamisches HTML](#)

Selektoren (Stylesheets)

- [Formatieren mit Stylesheets](#)

Server Side Includes

- [Erste serverseitige Techniken](#)

#config

- [Erste serverseitige Techniken](#)

#echo

- [Erste serverseitige Techniken](#)
- [Erste serverseitige Techniken](#)
- [Erste serverseitige Techniken](#)

#exec

- [Erste serverseitige Techniken](#)

#flastmod

- [Erste serverseitige Techniken](#)

#fsize

- [Erste serverseitige Techniken](#)

#include

- [Erste serverseitige Techniken](#)
- [Erste serverseitige Techniken](#)

Aufbau

- [Erste serverseitige Techniken](#)

CGI-Programme ausführen

- [Erste serverseitige Techniken](#)

Dateigröße einfügen

- [Erste serverseitige Techniken](#)

Dateiinhalte einfügen

- [Erste serverseitige Techniken](#)
- [Erste serverseitige Techniken](#)

Datum der letzten Änderung

- [Erste serverseitige Techniken](#)
- [Erste serverseitige Techniken](#)

Datusmausgabe konfigurieren

- [Erste serverseitige Techniken](#)

in HTML-Code einbinden

- [Erste serverseitige Techniken](#)

konfigurieren

- [Erste serverseitige Techniken](#)

modularer Seitenaufbau

- [Erste serverseitige Techniken](#)

Serverkonfiguration

- [Erste serverseitige Techniken](#)

Sicherheitsrisiken

- [Erste serverseitige Techniken](#)

Umgebungsvariablen einfügen

- [Erste serverseitige Techniken](#)

Serverseitige Techniken

- [Wozu JavaScript?](#)

ASP

- [Active Server Pages](#)

CGI

- [CGI und Perl](#)

JavaScript

- [Erste serverseitige Techniken](#)

PHP

- [PHP - Einführung](#)

Server Side Includes

- [Erste serverseitige Techniken](#)

setTimeout (JavaScript-Methode)

- [Animationen](#)

Sicherheit

CGI und Formulareingaben

- [CGI und Perl](#)

Java-Applets

- [Java-Applets](#)

Sichtbarkeit

- [Positionieren mit Stylesheets](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Skriptsprachen

- [Wozu JavaScript?](#)

Software

- [Von der Idee zum eigenen Web](#)

Adobe Photoshop

- [Von der Idee zum eigenen Web](#)

DreamWeaver

- [Von der Idee zum eigenen Web](#)
- [Von der Idee zum eigenen Web](#)

Editor

- [Von der Idee zum eigenen Web](#)

FrontPage

- [Von der Idee zum eigenen Web](#)
- [Von der Idee zum eigenen Web](#)

FTP_PRO

- [Von der Idee zum eigenen Web](#)

FTP-Programme

- [Von der Idee zum eigenen Web](#)

HTML-Kit

- [Von der Idee zum eigenen Web](#)

Microsoft Word

- [Von der Idee zum eigenen Web](#)

Notepad-Editor

- [Von der Idee zum eigenen Web](#)

Paint Shop Pro

- [Am Anfang war... HTML](#)

vi

- [Von der Idee zum eigenen Web](#)

Webserver

- [Von der Idee zum eigenen Web](#)

WS_FTP

- [Von der Idee zum eigenen Web](#)

Sonderzeichen

CGI

- [CGI und Perl](#)

HTML

- [Am Anfang war... HTML](#)
-

Perl

- [CGI und Perl](#)
- [CGI und Perl](#)
- [CGI und Perl](#)
- [CGI und Perl](#)

Sounddateien

- [Fortgeschrittenes und dynamisches HTML](#)

Sprachcodes

- [Fortgeschrittenes und dynamisches HTML](#)

SQL

- [Datenbankanbindung](#)

AUTO_INCREMENT

- [Datenbankanbindung](#)

CREATE TABLE

- [Datenbankanbindung](#)

Daten aktualisieren

- [Datenbankanbindung](#)

Daten lesen

- [Datenbankanbindung](#)

Daten löschen

- [Datenbankanbindung](#)

Daten schreiben

- [Datenbankanbindung](#)

DELETE

- [Datenbankanbindung](#)

DROP TABLE

- [Datenbankanbindung](#)

Fremdschlüssel

- [Datenbankanbindung](#)

INSERT INTO

- [Datenbankanbindung](#)

ORDER BY

- [Datenbankanbindung](#)

PRIMARY KEY

- [Datenbankanbindung](#)

SELECT

- [Datenbankanbindung](#)

SELECT COUNT

- [Datenbankanbindung](#)

SELECT DISTINCT

- [Datenbankanbindung](#)

sortieren

- [Datenbankanbindung](#)

Tabellen erstellen

- [Datenbankanbindung](#)

Tabellen löschen

- [Datenbankanbindung](#)

UPDATE

- [Datenbankanbindung](#)

SSI

- [Erste serverseitige Techniken](#)

Steuerelemente

- [Tabellen, Frames und Formulare](#)

Auswahlfeld

- [Tabellen, Frames und Formulare](#)

Beschriftung

- [Tabellen, Frames und Formulare](#)

Bild

- [Tabellen, Frames und Formulare](#)

Datei

- [Tabellen, Frames und Formulare](#)

Ereignisbehandlung

- [Tabellen, Frames und Formulare](#)

Ereignisverarbeitung

- [JavaScript und Formulare](#)

Kontrollkästchen

- [Tabellen, Frames und Formulare](#)

Optionsfeld

- [Tabellen, Frames und Formulare](#)

Schaltfläche

- [Tabellen, Frames und Formulare](#)

Textfeld

Stileigenschaften

- [Formatieren mit Stylesheets](#)

JavaScript

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

String (JavaScript-Klasse)

- [JavaScript- Grundkurs](#)

Strings

- [JavaScript- Grundkurs](#)
- [JavaScript- Grundkurs](#)

Stylesheets

- [Formatieren mit Stylesheets](#)

!important-Spezifizierer

- [Formatieren mit Stylesheets](#)

<style>-Tag

- [Formatieren mit Stylesheets](#)

alle Vorkommen eines HTML-Tags formatieren

- [Formatieren mit Stylesheets](#)

Browser

- [Formatieren mit Stylesheets](#)

Cascading

- [Formatieren mit Stylesheets](#)

CSS-Positionierung

- [Positionieren mit Stylesheets](#)

definieren

- [Formatieren mit Stylesheets](#)
- [Formatieren mit Stylesheets](#)

Definitionsreihenfolge

- [Formatieren mit Stylesheets](#)

eingebettete

- [Formatieren mit Stylesheets](#)

einzelne HTML-Elemente formatieren

- [Formatieren mit Stylesheets](#)

externe

- [Formatieren mit Stylesheets](#)

Größenangaben

- [Formatieren mit Stylesheets](#)

Hyperlinks

- [Formatieren mit Stylesheets](#)

importieren

- [Formatieren mit Stylesheets](#)

Inline-Stile

- [Am Anfang war... HTML](#)
- [Formatieren mit Stylesheets](#)

JavaScript

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Kommentare

- [Formatieren mit Stylesheets](#)

Listen

Medien-Typen

- [Formatieren mit Stylesheets](#)

selbst definierte Gruppen von HTML-Elementen formatieren

- [Formatieren mit Stylesheets](#)

Selektoren

- [Formatieren mit Stylesheets](#)
- [Formatieren mit Stylesheets](#)

Sichtbarkeit

- [Positionieren mit Stylesheets](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Spezifität

- [Formatieren mit Stylesheets](#)

Standard-Stylesheetsprache

- [Formatieren mit Stylesheets](#)

Stildefinitionen

- [Formatieren mit Stylesheets](#)

Stileigenschaften

- [Formatieren mit Stylesheets](#)

style-Attribut

- [Am Anfang war... HTML](#)
- [Formatieren mit Stylesheets](#)

Vererbung

- [Formatieren mit Stylesheets](#)

versus traditionelle Formatierung

- [Am Anfang war... HTML](#)

vor älteren Browsern verstecken

- [Formatieren mit Stylesheets](#)

Vorteile

- [Formatieren mit Stylesheets](#)

Suchmaschinen

- [Fortgeschrittenes und dynamisches HTML](#)

Anmeldung

- [Fortgeschrittenes und dynamisches HTML](#)

Rankingsystem

- [Fortgeschrittenes und dynamisches HTML](#)

Webroboter

- [Fortgeschrittenes und dynamisches HTML](#)

Swing

- [Java-Applets](#)

switch-Verzweigung

- [JavaScript- Grundkurs](#)

T

Tabellen

- [Tabellen, Frames und Formulare](#)

Breite

- [Tabellen, Frames und Formulare](#)

formatieren

- [Tabellen, Frames und Formulare](#)

Gitterlinien

- [Tabellen, Frames und Formulare](#)

Hintergründe

- [Formatieren mit Stylesheets](#)

Rahmen

- [Tabellen, Frames und Formulare](#)

Spaltengruppen

- [Tabellen, Frames und Formulare](#)

spezielle Stileigenschaften

- [Formatieren mit Stylesheets](#)

Überschrift

- [Tabellen, Frames und Formulare](#)

verschachteln

- [Tabellen, Frames und Formulare](#)

versus CSS-Positionierung

- [Positionieren mit Stylesheets](#)

Zeilen

- [Tabellen, Frames und Formulare](#)

Zellen

- [Tabellen, Frames und Formulare](#)

zentrieren

- [Tabellen, Frames und Formulare](#)

Tags

- [Am Anfang war... HTML](#)
- [Am Anfang war... HTML](#)

Text

- [Am Anfang war... HTML](#)

Absätze

- [Am Anfang war... HTML](#)

einrücken

- [Am Anfang war... HTML](#)

formatieren

- [Am Anfang war... HTML](#)
- [Formatieren mit Stylesheets](#)

Freiräume

- [Formatieren mit Stylesheets](#)

mit JavaScript ausgeben

- [Wozu JavaScript?](#)

Schrift

- [Formatieren mit Stylesheets](#)

Textausrichtung

- [Formatieren mit Stylesheets](#)

um Bilder

- [Positionieren mit Stylesheets](#)

Wortabstand

- [Formatieren mit Stylesheets](#)

Zeichenabstand

- [Formatieren mit Stylesheets](#)

Zeilenhöhe

- [Formatieren mit Stylesheets](#)

Zeilenumbrüche

- [Am Anfang war... HTML](#)

this

- [JavaScript- Grundkurs](#)
- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Threads

- [Animationen](#)

Thumbnails

- [Am Anfang war... HTML](#)

Ticker

- [Animationen](#)

Transparenz

- [Am Anfang war... HTML](#)

Typenumwandlung

- [JavaScript- Grundkurs](#)

U

Umgebungsvariablen

- [Erste serverseitige Techniken](#)
- [CGI und Perl](#)

undefined

- [JavaScript- Grundkurs](#)

Urheberrechte

- [Ein gelungener Webauftritt](#)

URLs

- [Fortgeschrittenes und dynamisches HTML](#)
 - absolute
 - [Fortgeschrittenes und dynamisches HTML](#)
 - Basisadresse
 - [Fortgeschrittenes und dynamisches HTML](#)
 - relative
 - [Fortgeschrittenes und dynamisches HTML](#)

Usability

- [Ein gelungener Webauftritt](#)

V

Variablen

- [JavaScript- Grundkurs](#)
 - globale
 - [JavaScript- Grundkurs](#)
 - lokale
 - [JavaScript- Grundkurs](#)

Verdeckung

- [JavaScript- Grundkurs](#)

Vererbung

- Stylesheets
 - [Formatieren mit Stylesheets](#)

Vergleiche

- [JavaScript- Grundkurs](#)

Versionsinformation

- [Am Anfang war... HTML](#)

Videos

- [Fortgeschrittenes und dynamisches HTML](#)

WAMP

- [PHP - Einführung](#)

Webcrawler

- [Fortgeschrittenes und dynamisches HTML](#)

Webdesign

- [Von der Idee zum eigenen Web](#)

Webroboter

- [Fortgeschrittenes und dynamisches HTML](#)

Webs

auf lokalem Server

- [Von der Idee zum eigenen Web](#)

Bildergalerien

- [Tabellen, Frames und Formulare](#)

Dokumentenhierarchie

- [Fortgeschrittenes und dynamisches HTML](#)

Formulare

- [CGI und Perl](#)

Gästebücher

- [CGI und Perl](#)

Generationen

- [Von der Idee zum eigenen Web](#)

hochladen auf Server

- [Von der Idee zum eigenen Web](#)

Inhaltsverzeichnisse

- [Am Anfang war... HTML](#)
- [Am Anfang war... HTML](#)

Internet Service Provider wählen

- [Von der Idee zum eigenen Web](#)

Navigationsleisten

- [Am Anfang war... HTML](#)

Planung und Erstellung

- [Von der Idee zum eigenen Web](#)

Serverseitige ImageMaps

- [CGI und Perl](#)

Tabellen-Layouts

- [Tabellen, Frames und Formulare](#)

Usability

- [Ein gelungener Webauftritt](#)
- verschieben
- [Fortgeschrittenes und dynamisches HTML](#)
- Webseiten testen
- [Von der Idee zum eigenen Web](#)
- Zeitungslayout
- [Tabellen, Frames und Formulare](#)
- Zielsetzung
- [Von der Idee zum eigenen Web](#)

Webseiten

Absätze

- [Am Anfang war... HTML](#)

Animationen

- [Animationen](#)

Aufzählungen

- [Am Anfang war... HTML](#)

automatisch aktualisieren

- [Fortgeschrittenes und dynamisches HTML](#)

automatisch wechseln

- [Fortgeschrittenes und dynamisches HTML](#)

Basisadresse

- [Fortgeschrittenes und dynamisches HTML](#)

Bilder

- [Am Anfang war... HTML](#)

Datum ausgeben

- [JavaScript- Grundkurs](#)

Datum der letzten Änderung

- [Erste serverseitige Techniken](#)

Datum einblenden

- [Wozu JavaScript?](#)

Datum einfügen

- [Erste serverseitige Techniken](#)

Definitionen

- [Am Anfang war... HTML](#)

dynamisch erzeugen

- [CGI und Perl](#)

dynamische Listen

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Elemente logisch gruppieren

- [Am Anfang war... HTML](#)

Elemente positionieren

- [Positionieren mit Stylesheets](#)

Elemente überlagern

- [Positionieren mit Stylesheets](#)

Farben

- [Am Anfang war... HTML](#)

Flash-Animationen einbinden

- [Animationen](#)

Frames

- [Tabellen, Frames und Formulare](#)

Hintergrundbilder

- [Am Anfang war... HTML](#)

horizontale Linien

- [Am Anfang war... HTML](#)

Hyperlinks

- [Am Anfang war... HTML](#)

Informationen für Webseiten

- [Fortgeschrittenes und dynamisches HTML](#)

Java-Applets

- [Java-Applets](#)

JavaScript-Code

- [Wozu JavaScript?](#)

Listen

- [Am Anfang war... HTML](#)

modularer Aufbau

- [Erste serverseitige Techniken](#)

Nummerierungen

- [Am Anfang war... HTML](#)

Rollover-Schalter

- [Mit JavaScript auf HTML-Elemente zugreifen](#)

Schriften

- [Am Anfang war... HTML](#)

Seitenübergänge

- [Fortgeschrittenes und dynamisches HTML](#)

Tabellen

- [Tabellen, Frames und Formulare](#)

Text

- [Am Anfang war... HTML](#)

Textpassagen auszeichnen

- [Am Anfang war... HTML](#)

Titel

- [Am Anfang war... HTML](#)

Überschriften

- [Am Anfang war... HTML](#)

Webserver

- [Von der Idee zum eigenen Web](#)
- [Der Apache-Server](#)

Apache-Server

- [Von der Idee zum eigenen Web](#)
- [PHP - Einführung](#)

CGI

- [CGI und Perl](#)

Dokumentenverzeichnis

- [Von der Idee zum eigenen Web](#)

Installation

- [Von der Idee zum eigenen Web](#)
- [Der Apache-Server](#)

Internet Information Server

- [Active Server Pages](#)

localhost

- [Von der Idee zum eigenen Web](#)

lokaler

- [Von der Idee zum eigenen Web](#)

OmniHttpd

- [Von der Idee zum eigenen Web](#)
- [Der Apache-Server](#)

Personal Web Server

- [Von der Idee zum eigenen Web](#)
- [Active Server Pages](#)
- [PHP - Einführung](#)

Server Side Includes

- [Erste serverseitige Techniken](#)

Webseiten testen

- [Von der Idee zum eigenen Web](#)

Xitami

- [Von der Idee zum eigenen Web](#)

Websoftware

- [Von der Idee zum eigenen Web](#)

while-Schleife

- [JavaScript- Grundkurs](#)

window (JavaScript-Objekt)

- [Wozu JavaScript?](#)
- [JavaScript- Grundkurs](#)

Windows-Konsole

- [Java-Applets](#)

with

- [JavaScript- Grundkurs](#)

WS_FTP

- [Von der Idee zum eigenen Web](#)

X

XHTML

- [XML und XHTML](#)
aus HTML
 - [XML und XHTML](#)
- Groß- und Kleinschreibung
 - [XML und XHTML](#)
- Motivation
 - [XML und XHTML](#)
- Wurzelement
 - [XML und XHTML](#)

Xitami

- [Von der Idee zum eigenen Web](#)

XML

- [XML und XHTML](#)
#PCDATA
 - [XML und XHTML](#)
- <!ELEMENT>
 - [XML und XHTML](#)
- Attribute
 - [XML und XHTML](#)
- CDATA
 - [XML und XHTML](#)
- CSS
 - [XML und XHTML](#)
- DTD
 - [XML und XHTML](#)
- einbauen
 - [XML und XHTML](#)

Elemente

- [XML und XHTML](#)

Entity

- [XML und XHTML](#)

Expat

- [XML und XHTML](#)

im Internet Explorer

- [XML und XHTML](#)

Motivation

- [XML und XHTML](#)

Parser

- [XML und XHTML](#)
- [XML und XHTML](#)

Tags

- [XML und XHTML](#)

wohlgeformt

- [XML und XHTML](#)

Wurzelement

- [XML und XHTML](#)

XSL

- [XML und XHTML](#)

XSL

- [XML und XHTML](#)

Z

Zeilenumbrüche

- [Am Anfang war... HTML](#)
in Browser erzwingen
 - [Am Anfang war... HTML](#)
- in JavaScript
 - [Wozu JavaScript?](#)

Zeitungslayout

- [Tabellen, Frames und Formulare](#)

Zufallszahlen

- [JavaScript- Grundkurs](#)

Wir sind an Ihrer Meinung interessiert!

Als Teil unseres Bestrebens, Bücher höchster Qualität zu produzieren, ist SAMS an Ihrer Meinung interessiert.

Sie können Ihre Kommentare, Ideen oder Vorschläge zur Verbesserung zukünftiger Ausgaben an die unten angegebene Adresse oder per Fax an 089-46003-330 schicken.

Online-Verbindung bekommen Sie über das Internet <http://www.mut.de>. Schon jetzt vielen Dank – Ihre Kommentare werden uns dabei helfen, weiterhin die besten Computerbücher herauszugeben.

SAMS

Markt+Technik

[Erik Franz](#)

Hans-Pinsel-Str. 9b

85540 Haar bei München

Fragen zur HTML-Version?

Markt+Technik

[Harry Jörg](#)

Hans-Pinsel-Str. 9b

85540 Haar bei München

Vorwort

Webseiten zu erstellen, ist heute etwas ganz anderes als noch vor wenigen Jahren. Während es früher vordringlich darum ging, überhaupt mit einem Webangebot oder einer privaten Homepage im Web präsent zu sein, interessieren sich heute mehr und mehr Webseitenbesitzer für Fragen des Webdesigns und der Erstellung dynamischer und interaktiver Webseiten. Die Ansprüche sind gestiegen. Dieses Buch soll Ihnen helfen, den allgemeinen und Ihren eigenen Ansprüchen gerecht zu werden.

An wen sich dieses Buch richtet

- An Einsteiger

Obwohl dieses Buch nicht für Einsteiger als Hauptzielgruppe konzipiert wurde, sollte es für sie dennoch bestens geeignet sein, da es verständlich in alle wichtigen Fragen einführt: von der Erstellung der ersten Webseite über Grundregeln des Webdesigns und die Webprogrammierung mit JavaScript bis hin zum Veröffentlichen und Bekanntmachen der fertigen Seiten.

Da weder HTML noch JavaScript sonderlich schwer zu erlernen sind, werden Sie mit diesem Buch sicherlich gut zurecht kommen, zumal alle wichtigen Konzepte, wie gesagt, ausführlich und verständlich erklärt werden. Allerdings gehen wir in diesem Buch etwas schneller voran, als in einem Buch, das nur HTML oder JavaScript gewidmet ist. Wenn also an der einen oder anderen Stelle zu viele Informationen auf einmal auf Sie einströmen sollten, legen Sie einfach eine kleine Pause ein und geben Sie dem Gelernten Zeit sich zu setzen (die Quizfragen und Übungen können Ihnen dabei helfen). Dafür können Sie das Buch auch als Nachschlagewerk und zur Fortbildung nutzen.

Schließlich bleibt noch zu erwähnen, dass sich dieses Buch streng an den aktuellen HTML/XHTML-Standards und den neuesten Browsern orientiert, so dass Sie keine Gefahr laufen, sich Techniken anzueignen, die in kürzester Zeit vielleicht schon veraltet sind.

- An Fortgeschrittene

Wenn Sie schon erste eigene Webseiten erstellt haben, vielleicht auch schon erste Gehversuche mit JavaScript oder gar einer serverseitigen Programmiersprache unternommen haben, dann dürfte dieses Buch ideal für Sie sein.

Sie können dieses Buch zum Ausbau und zur Vervollständigung Ihrer Kenntnisse nutzen, Sie können Ihr Wissen auf den letzten Stand bezüglich Unterstützung der aktuellen Browser (einschließlich Netscape 6) und Standards (HTML 4, XHTML) bringen, Sie können sich in einen großen Kreis von Themen rund um die Erstellung von Webseiten einarbeiten, Sie können das Buch zur Fortbildung und als Referenz nutzen.

- An Experten

Angesichts der vielfältigen Techniken, die heutzutage in Zusammenhang mit der Erstellung von Webseiten zum Einsatz kommen, ist es selbst für professionelle Webdesigner und -programmierer kaum noch möglich, in allen Bereichen gleichermaßen kompetent zu sein.

Mit Hilfe dieses Buches können Sie Wissenslücken schließen und sich in die verschiedensten Techniken einarbeiten: wie HTML 4, XHTML, CSS, JavaScript, Java, CGI, Flash-Animationen, GIF-Animationen, DHTML, DOM, ASP, PHP, Datenbankbindung, XML, Perl.

Was Sie von diesem Buch erwarten dürfen

Die erste Woche soll die Grundlage für Ihre Tätigkeit als Webdesigner/ Webprogrammierer legen. Hier führen wir Sie in HTML, CSS sowie in Fragen des Webdesigns oder der Veröffentlichung von Webseiten im Web oder auf einem lokalen Webserver ein. Die damit verbundenen Konzepte werden ausführlich, verständlich und praxisnah erklärt, einfachere Themen werden nicht zu sehr ausgebreitet, so dass sich die Woche auch gut als Referenz zu HTML und CSS verwenden lässt.

Die zweite Woche ist der Programmierung mit JavaScript gewidmet. Wir wollen Sie in dieser Woche aber nicht nur mit JavaScript bekannt machen, wir wollen Sie auch am Beispiel von JavaScript in die Programmierung allgemein einführen. Viele Konzepte, die sich in praktisch allen heutigen, strukturierten Programmiersprachen wiederfinden,

werden in dieser Woche anhand von JavaScript vorgestellt und ausführlich beschrieben. Diese Woche bildet damit auch die Grundlage für die dritte Woche, in der Sie eine Reihe von weiteren Programmiersprachen für die Unterstützung von Webseiten (VBScript, Java, Perl, PHP) kennen lernen werden und in denen Sie die anhand von JavaScript vorgestellten Konzepte unschwer wiedererkennen werden.

Das wir dabei die Erstellung dynamischer Webseiten mit Hilfe von JavaScript und DHTML nicht vernachlässigen, dürfte selbstverständlich sein. Für Leser, die sich bereits mit JavaScript auskennen, dürfte besonders interessant sein, dass wir in allen Kapiteln auf browserunabhängige Implementierungen achten, die im Internet Explorer und im neuen Netscape 6-Browser lauffähig sind. Auch auf den immer noch weit verbreiteten Netscape Navigator 4 gehen wir ein - ohne jedoch ihn oder seine, durch den Netscape 6-Browser mittlerweile überholten Techniken zu sehr in den Vordergrund zu rücken.

In der dritten Woche führen wir Sie in eine Reihe von fortgeschrittenen, größtenteils serverseitigen Techniken zur Unterstützung von Webseiten ein - mit zwei eigenen Kapiteln zu den Themen Animation und Datenbankbindung.

Zum guten Schluss möchten wir noch erwähnen, dass wir großen Wert auf Verständlichkeit, Ausführlichkeit, Aktualität und Praxisnähe gelegt haben. Erstaufgaben haben allerdings die Eigenheit, die Bemühungen ihrer Autoren schon einmal zu konterkarieren. Sollten Ihnen Fehler auffallen, scheuen Sie sich nicht uns zu mailen. Wir werden diese dann auf dem Markt+Technik-Server unter www.mut.de veröffentlichen.

Dirk Louis, dirklouis@cs.com

Christian Wenz, Christian_Wenz@compuserve.com

❖·Kapitel Inhalt Index **SAMS** ❖· Top Kapitel·❖

© [Markt+Technik Verlag](http://www.mut.de), ein Imprint der Pearson Education Deutschland GmbH

Tag 1

Von der Idee zum eigenen Web

Die meisten Leser dieses Buches werden sicherlich schon über ihr eigenes Web im Internet verfügen und sich für einen Webeditor ihrer Wahl entschieden haben. Wenn dem so ist, dann wollen wir daran gar nichts ändern. Ziel dieses Buches ist es, Sie mit den grundlegenden Techniken des dynamischen Web Publishing vertraut zu machen und Sie mit interessanten Tipps und (möglichst) kompletten Lösungen zur Ausgestaltung und Verbesserung Ihres Webs zu versorgen - und nicht Ihnen vorzuschreiben, welchen Webeditor Sie verwenden, welche Webtools Sie nutzen und auf welchen Server Sie veröffentlichen sollen. Anders ausgedrückt: Die hier vorgestellten Technologien sind unabhängig von irgendwelchen Webeditoren und weitgehend unabhängig vom verwendeten Webserver (der allerdings unter Umständen entsprechend konfiguriert sein muss, siehe Abschnitt 1.4 und Hinweise in den entsprechenden Kapiteln).

Da es aber auch zweifelsohne Leser geben wird, für die dieses Buch der Einstieg in die Webseitenerstellung überhaupt darstellt, wollen wir kurz rekapitulieren, welche Schritte, Überlegungen und Werkzeuge zur Erstellung und Veröffentlichung von Webseiten nötig sind.



Da wir uns nicht von irgendwelcher Websoftware abhängig machen wollen, werden wir in diesem Buch alles in Handarbeit machen, was meist bedeutet, dass wir den HTML-Code der Webseiten direkt bearbeiten. Dem Anfänger mag dies auf den ersten Blick vielleicht etwas rückschrittlich und archaisch vorkommen, schließlich gibt es heutzutage genügend leistungsfähige Webeditoren, mit denen man Webseiten aufsetzen kann, ohne dass man auch nur einmal mit HTML, JavaScript oder CGI in Berührung kommt. Das Problem ist nur, dass dieser Form des Webdesigns enge Grenzen gesetzt sind, und der angehende Webautor über kurz oder lang einsehen muss, dass er ohne fundierte Kenntnisse in HTML, JavaScript, etc. nicht weiterkommt. Aus diesem Grunde möchten wir Sie bitten, erst einmal auf die besonderen »Annehmlichkeiten« Ihres Webeditors zu verzichten und den Ausführungen in diesem Buch zu folgen, um zu verstehen, wie die betreffenden Technologien arbeiten und wie sie eingesetzt werden. Danach können Sie selbstverständlich frei entscheiden, ob und inwieweit Sie die spezielle Unterstützung Ihres Webeditors nutzen wollen.

1.1 Idee und Design

Die Entstehung einer Webseite beginnt mit einer Idee. Oft genug ist dies eine nur sehr vage Idee, und der erste Schritt besteht daher üblicherweise darin, die Idee zu konkretisieren - und zwar unter dreierlei Gesichtspunkten:

- Welches Ziel wird mit dieser Webseite angestrebt und wie sieht die Zielgruppe aus?
- Welchen Inhalt soll die Webseite transportieren?
- Wie lässt sich die Webseite technisch realisieren?

Keiner dieser drei Aspekte kann isoliert von den anderen betrachtet werden, und alle zusammen bestimmen sie das Design der Webseite, das seinerseits wieder auf Inhalt, Zielsetzung und Technik rückwirkt.

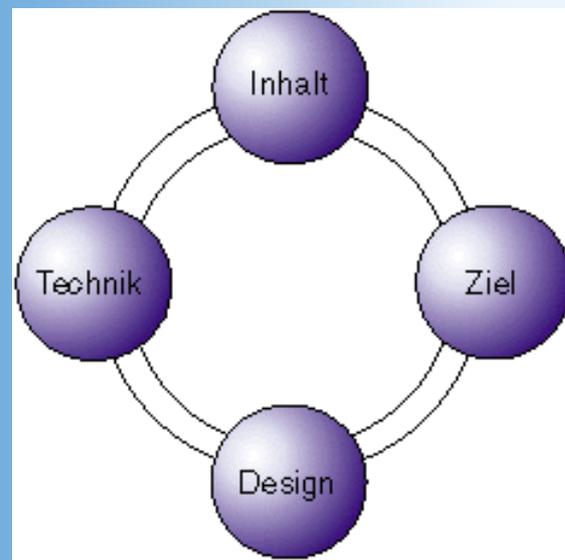


Abbildung 1.1: Aspekte modernen Webdesigns

Zielsetzung

Die wichtigsten Zielsetzungen für Webseiten sind: Präsentation, Information und Unterhaltung, Kommerz und Kommunikation. Private Homepages dienen beispielsweise üblicherweise der Präsentation ihres Inhabers, Webseiten behördlicher Institute und gemeinnütziger Vereine dienen der Information und Webseiten von Firmen sind typischerweise dazu gedacht, den Umsatz der Firma zu steigern. Webseiten, die hauptsächlich der Kommunikation dienen, gibt es nur wenige. Dafür verfügen praktisch alle Webseiten über das eine oder andere Kommunikationselement (beispielsweise den obligatorischen EMail-Link zum Webseitenbetreiber oder Formulare zum Übersenden von Daten).

Selbstverständlich kann eine Webseite oder ein Web (eine Gruppe zusammengehörender Webseiten, siehe auch Abschnitt 1.2, »Von der Webseite zum Web«) gleichzeitig mehrere Zielsetzungen verfolgen. Kommerzielle Ziele beispielsweise können überhaupt nur in Verbindung mit Präsentation, Information und/oder Kommunikation verwirklicht werden.

| | |
|---------------------------|--|
| Kommerz und Präsentation | Firmen können mit Hilfe eines eigenen Webs ihren Bekanntheitsgrad steigern sowie Kundennähe und Servicefreundlichkeit demonstrieren. |
| Kommerz und Kommunikation | Firmen, die Produkte erstellen oder mit Produkten handeln, die für den Postversand geeignet sind, werden ihre Webs mit einem elektronischen Bestellsystem ausstatten.

Aber nicht nur Waren, auch Dienstleistungen können übers Internet angeboten werden. |
| Kommerz und Information | Wer durch Werbung auf seinen Webseiten Geld verdienen möchte, muss kostenfreie Informationen und Dienstleistungen anbieten, die seine Seiten attraktiv machen und viele Besucher anziehen. |

Inhalt und Design

Der grundlegende Inhalt einer Webseite (oder eines Webs) wird bereits weitgehend durch ihre Zielsetzung vorgegeben. Eine Firma wird das eigene Web nutzen, um sich selbst zu präsentieren, ihre Produkte (oder Dienstleistungen) vorzustellen und eventuell elektronische Bestellungen anzunehmen. Privatleute nutzen die eigene Homepage meist, um sich selbst darzustellen, ihre Interessengebiete vorzustellen oder das Internet in uneigennütziger (manchmal auch gesponserter) Weise auf irgendeine Weise zu bereichern.

Ist man sich über Inhalt und Zielsetzung im Klaren, beginnt die Design-Phase. Je professioneller die Webseite sein soll, um so mehr Zeit und Arbeit sollte man in das Design stecken. Ein gutes Design verleiht einer Webseite einen unverkennbaren, eigenen Stil, es basiert auf einem klaren Konzept, es ordnet die Elemente der Webseite, es überzeugt den Besucher. Ein Design kann schlicht oder ausgefallen, konservativ oder poppig, sachlich oder verspielt sein. Doch nie darf es der Funktion der Webseite entgegenstehen. Wir werden dazu noch einige Beispiele sehen.

Überhaupt wird uns das Thema Design durch das gesamte Buch hindurch immer wieder beschäftigen. Schließlich wäre es reiz- und sinnlos, würden wir Sie mit den technischen Möglichkeiten des dynamischen Webpublishings vertraut machen, ohne dabei auch das eine oder andere Wort über die sinnvolle Verwendung der betreffenden dynamischen Elemente zu verlieren. Die folgenden, grundlegenden Regeln können Sie sich aber schon einmal einprägen:

- Webseiten, insbesondere Homepages, sollten keine allzu persönlichen oder gar peinlichen Inhalte haben.
- Webseiten sollten keine Beschimpfungen, Verunglimpfungen, keine Volksverhetzung und keine kriminellen Inhalte aufweisen.
- Gutes Design ist auch eine Kostenfrage. Für gutes Design braucht man entsprechende Software und Zeit. Apropos Zeit: Denken Sie bei der Einbindung von Bildern und anderen multimedialen Elementen auch an die Besucher ihres Webs und versuchen Sie, die Download-Zeiten niedrig zu halten.
- Überfrachten Sie Ihre Webseiten nicht mit zuviel dynamischen Inhalten und Spielereien.
- Veröffentlichen Sie keine illegalen Inhalte, klauen Sie keine Copyright-geschützten Ressourcen (Bilder, Videos, etc.).

- Entwickeln Sie einen eigenen Stil.

Oder um es mit den Worten Shakespeares zu sagen:

Und diese Regeln präg´ in dein Gedächtnis:

Gib den Gedanken, die du hegst, nicht Zunge,

Noch einem ungemäßigten die Tat.

Freundlich sei, doch keineswegs gemein.

...

Die Kleidung kostbar, wie´s dein Beutel kann,

Doch nicht ins Grillenhafte; reich , nicht bunt:

Denn es verkündet oft die Tracht den Mann,

Und die vom ersten Rang und Stand

Sind darin ausgesucht und edler Sitte.

Kein Borger sei und auch Verleiher nicht;

sich und den Freund verliert das Darlehn oft,

Und borgen stumpft der Wirtschaft Spitze ab.

Dies über alles: sei dir selber treu.

Shakespeare, Hamlet

Technische Realisierung

So langsam sollten wir darangehen, unsere Ideen auch umzusetzen.

Das Grundgerüst der Webseiten wird in HTML realisiert. HTML (HyperText Markup Language) ist eine Beschreibungssprache, mit deren Hilfe wir (in gewissen Grenzen) festlegen können, wie unsere Webseiten in den verschiedenen Webbrowsern dargestellt werden sollen. Mehr dazu in den folgenden Kapiteln.

Bilder und grafische Elemente sind für das Design der meisten Webseiten von außerordentlicher Bedeutung. Ohne den Wert des vorliegenden Buches schmälern zu wollen, möchte ich doch anmerken, dass ein interessantes grafisches Design mehr zur Attraktivität eines Webs beitragen kann als die Implementierung eines Gästebuches oder irgendwelcher

dynamischer Effekte. Allerdings benötigt man für die Erstellung und Bearbeitung von Bildern, Hintergründen, Grafiken, etc. je nach angestrebtem Perfektionsgrad viel Zeit, grafisches Gespür (oder eine Ausbildung als Grafiker) und entsprechende Software (Programme zur Bildbearbeitung, zum Scannen, zur Erstellung von GIF-Animationen, zur Schriftgenerierung, etc.).

Dynamische Elemente, wie Zugriffszähler, Gästebücher, Formulare, Animationen, Spiele, etc., können mit HTML allein nicht realisiert werden. Sie basieren auf Programmcode, der entweder auf Seiten des Clients (durch Vermittlung des Browsers) oder auf Seiten des Servers (dem Rechner, von dem die Webseiten herkommen) ausgeführt wird. Wie man für Webseiten programmiert und wie man typische dynamische Elemente (beispielsweise Gästebücher) realisiert, ist Thema dieses Buches.

1.2 Aufsetzen des Codes - der HTML-Editor

Die Idee, die hinter der Beispielwebseite aus diesem Kapitel steht, ist ziemlich einfach: Wir brauchen nichts weiter als eine schlichte Webseite zum Testen unserer Entwicklungswerkzeuge und der Server-Verbindung:

Listing 1.1: testseite.html - Unsere HTML-Testseite

```
<html>
<head>
<title>Testseite</title>
</head>
<body>
<p><b>Hallo</b> von der Testseite !</p>
</body>
</html>
```

Diese Webseite besteht aus dem (vereinfachten) HTML-Grundgerüst (siehe Kapitel 2), einem Seitentitel, der üblicherweise in den Titelleisten der Browser angezeigt wird, und einem kleinen Begrüßungstext: »**Hallo** von der Testseite !«. Damit das Wort »Hallo« fett dargestellt wird, haben wir es in die HTML-Tags ` ... ` eingeschlossen.

Da der HTML-Code der Webseiten einfacher ASCII-Text ist, braucht man zum Erstellen einer solchen Webseite keine großartigen Hilfsmittel, ein simpler Texteditor (beispielsweise der Notepad-Editor von Windows, **Start/Programme/Zubehör/Editor**, oder der **vi** unter Linux) reichen vollkommen aus.

1. Rufen Sie einen Texteditor auf, der Dateien als reinen ASCII-Text speichern kann.
2. Tippen Sie den Quellcode aus Listing 1.1 ein.
3. Speichern Sie die Datei in einem beliebigen Verzeichnis unter dem Namen *testseite.html*.



*Viele Leser haben Probleme beim Abspeichern von Dateien mit dem Notepad-Editor von Windows, weil dieser standardmäßig alle Dateien mit der Extension .txt abspeichert. Um dies zu verhindern, müssen Sie im Dialogfeld **Speichern unter** als Dateityp den Eintrag **Alle Dateien (*.*)** auswählen und dann den Dateinamen mit der gewünschten Extension (in unserem Fall also .html) angeben. Ansonsten kann man die Dateierweiterung aber auch im Windows Explorer korrigieren (Befehl **Datei/Umbenennen**). Ein anderer Trick ist, den Dateinamen samt Extension in Anführungszeichen einzugeben: »webseite.html«.*

Spezielle Webeditoren

Wenn sie auf Ihrem PC einen ordentlich formatierten Brief oder ein Textdokument (beispielsweise ein Buchmanuskript) aufsetzen wollen, benötigen Sie dazu eine spezielle Textverarbeitungssoftware, mit der man Schriftarten auswählen, Text einrücken, Bilder einfügen und andere Dinge machen kann. Wenn Sie eine professionell gestylte Webseite erstellen wollen, brauchen Sie dazu nur einen einfachen ASCII-Texteditor, denn sämtliche Formatierungsbefehle werden als HTML-Code direkt in die Textdatei geschrieben. Dieses Verfahren hat allerdings nicht nur Vorteile, es hat auch Nachteile:

- Wenn Sie eine Textpassage fett darstellen wollen, können Sie nicht einfach den Text markieren und dann in Ihrer Textverarbeitungssoftware auf den Schalter für Fettdruck klicken (oder einen entsprechenden Menübefehl aufrufen). Sie müssen wissen, welches HTML-Tag für Fettdruck verantwortlich ist, und den Text in dieses Tag einschließen.
- Sie sehen beim Aufsetzen Ihres Textes immer nur den unformatierten HTML-Code. Um das Ergebnis Ihrer Formatierungs- und Layoutbefehle zu überprüfen, müssen Sie die HTML-Seite speichern und in einen Browser laden.

Hier schaffen spezielle Webeditoren wie FrontPage oder DreamWeaver Abhilfe. Mit diesen Editoren arbeiten Sie wie in einem »normalen« Textverarbeitungssystem und der Editor setzt im Hintergrund alle Formatierungen in HTML-Code (den man selbstverständlich auch anschauen und direkt bearbeiten kann). Wir werden in diesem Buch zwar ausschließlich auf HTML-Ebene operieren, doch soll Sie dies nicht davon abhalten, mit Ihrem favorisierten Webtool zu arbeiten. Zumal professionelle Webeditoren meist auch bei der Organisation und Erstellung ganzer Webs helfen.

Von der Webseite zum Web

Der gemeine Web-Neuling, begnügt sich üblicherweise erst einmal damit, eine einfache einzelne Webseite, seine private Homepage, ins Internet zu stellen. Er macht damit das Gleiche, was man auch aus der Entwicklung des menschlichen Fötus kennt: die individuelle Entwicklung (Embryogenese/Webdesign-Ausbildung) vollzieht die historische Entwicklung (Evolution/Design im WWW) nach.

Früher bestanden nämlich fast alle Homepages aus einer einzigen Webseite mit ein wenig Text, dem Konterfei des Inhabers und Links zu anderen Websites im WWW. Eine solche Homepage umfasste zwei Dateien, die HTML-Datei und eine Bilddatei für das Konterfei, die beide in einem Verzeichnis auf dem Webserver untergebracht wurden.

Der nächste Schritt war die Aufteilung des Inhalts auf mehrere Webseiten, die untereinander durch relative Links verbunden waren. Der klassische Aufbau einer solchen Homepage (man spricht in diesem Zusammenhang auch von Webseiten der 2. Generation) sah so aus, dass man auf einer Startseite begrüßt wurde und von dort über relative Links zu den weiteren Seiten des Webs verzweigen konnte (meist waren es drei untergeordnete Seiten: zur Vorstellung der eigenen Hobbys, zur Präsentation der privaten Fotoausstellung und zur Empfehlung von Links zu anderen interessanten Websites). Ein solcher Webauftritt, der aus mehreren zusammengehörenden Webseiten besteht, bezeichnet man als Web. Auf dem Webserver sind alle Dateien des Webs in einem Verzeichnis untergebracht (gegebenenfalls verteilt auf verschiedenen Unterverzeichnisse).

Heute sind viele Webs - private wie firmeneigene - wie Erlebnisparks organisiert: Sie besitzen einen speziellen Eingang, führen den Besucher ausgehend von der Hauptseite über relative Links im Web herum (wobei der Besucher selbst wählen kann, welchen Links er folgen, welche Attraktionen er anschauen möchte) und verabschieden den Besucher mit einer speziellen Seite (die auch schon einmal gleich der Hauptseite sein kann).

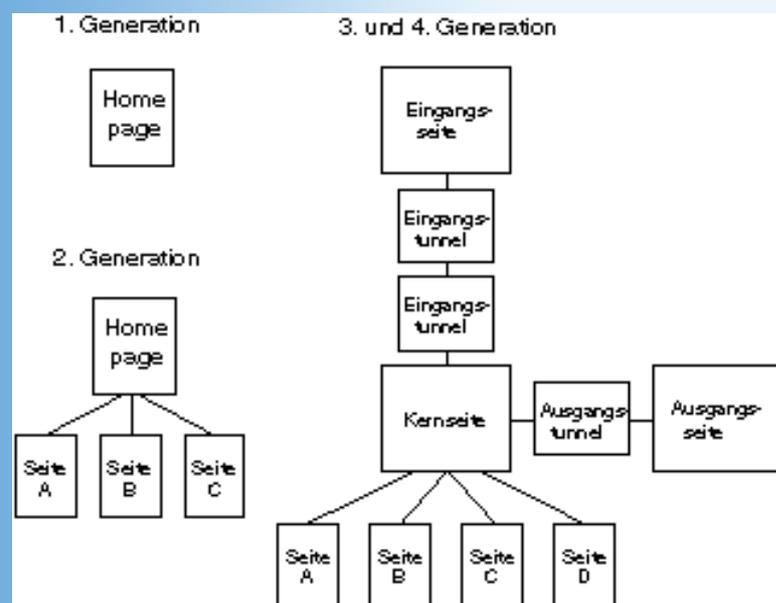


Abbildung 1.2: Generationen des Webdesigns

Je größer ein Web ist und um so mehr Dateien zu dem Web gehören (HTML-Dateien, Bild- und Sounddateien, Skriptdateien, Applets, etc.), um so mehr Gedanken muss man sich um die Einrichtung einer passenden Navigationsstruktur und die Organisation der Dateien auf der Festplatte machen. Professionelle Webeditoren, wie zum Beispiel FrontPage oder Dreamweaver) unterstützen den Webdesigner auch bei dieser Arbeit.

1.3 Webseiten im Browser testen

Kommen wir zurück zu unserer Beispielwebseite, *testseite.html*, die wir im vorangehenden Abschnitt erstellt haben.

4. Laden Sie die Webseite in Ihren Browser (beispielsweise den Netscape Navigator oder den Internet Explorer). Wenn auf Ihrem System die Datei-Extension *.html* mit Ihrem bevorzugten Browser als Standardverarbeitungsprogramm verbunden ist, brauchen Sie die HTML-Datei nur in Ihrem Datei-Manager (Windows Explorer unter Microsoft Windows oder Vorlage unter KDE/LINUX) anzuklicken (einfach oder doppelt). Ansonsten rufen Sie Ihren Browser auf und laden Sie die Datei über den entsprechenden Menübefehl oder durch Eingabe des Verzeichnispfades und des Dateinamens.

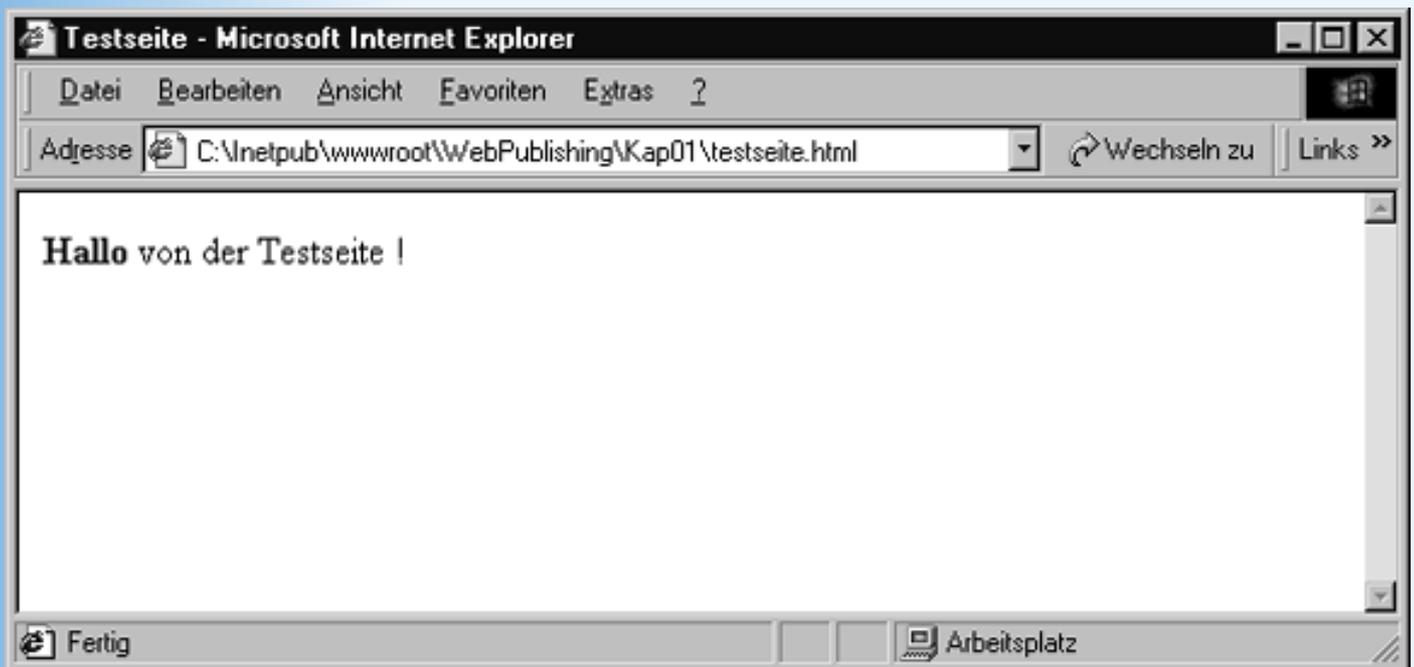


Abbildung 1.3: Unsere Testseite im Internet Explorer

Im Browser können Sie drei wichtige Dinge testen:

- Haben Sie korrekten HTML-Code aufgesetzt?

Wenn die HTML-Seite ohne Fehlermeldung geladen und angezeigt wird, ist dies schon einmal ein gutes Zeichen. Aber Achtung! Die meisten Browser sind äußerst fehlertolerant und akzeptieren auch HTML-Code, der nicht hundertprozentig korrekt oder gar verstümmelt ist. Prüfen Sie also auf alle Fälle, ob alle Elemente vorhanden und wie gewünscht formatiert sind.

- Sieht die Webseite so aus, wie Sie sich die Seite vorgestellt haben?

Viele Gründe können dafür verantwortlich sein, dass dies nicht der Fall ist. Wenn Sie den HTML-Code der Webseite in einem einfachen Texteditor aufsetzen, ist das größte Problem der HTML-Code selbst. Mit Hilfe der HTML-Tags können Sie zwar festlegen, wie die verschiedenen Elemente und Textbausteine auf der Webseite angeordnet und formatiert werden sollen, aber den Effekt der HTML-Tags sehen Sie erst, wenn Sie die Webseite in den

Browser laden.

Aber auch wenn Sie einen speziellen Webeditor wie FrontPage oder Dreamweaver verwenden, der den HTML-Code direkt umsetzt, kann es Unterschiede zwischen dem Erscheinungsbild in Ihrem Webeditor und Ihrem Browser geben. Der Grund hierfür ist, dass viele HTML-Tags nur grobe Hinweise auf die gewünschte Formatierung geben und dem Browser bei der Umsetzung entsprechend viel Freiheiten einräumen. So kommt es, dass ein und dieselbe HTML-Seite in verschiedenen Browsern unterschiedlich dargestellt wird. Man kann (muss) dies akzeptieren oder eine andere Form der Formatierung ausprobieren. Auf jeden Fall sollte man es sich aus diesem Grunde angewöhnen, die Darstellung der eigenen Webseiten in mehreren Browsern (zumindest Netscape Navigator und Internet Explorer) zu begutachten.

Schließlich kann es vorkommen, dass die Webseite im Browser zwar so erscheint, wie Sie es beabsichtigt hatten, Sie aber feststellen, dass Sie dieses Design nicht befriedigt.

- Sieht die Webseite auch dann noch gut aus, wenn Sie die Abmaße des Browser-Fensters verändern?

Es wurde bereits im vorangehenden Abschnitt angesprochen: der HTML-Code gibt das Layout und die Formatierung der Webseiten nur grob vor und überlässt die konkrete Umsetzung dem Browser. Dazu gehört auch, dass der Browser gewisse Freiheiten bei der Anordnung der verschiedenen Elemente (Bilder, Textbausteine) hat, genauer gesagt, er kann die Anordnung der Elemente an seine eigene Fensterbreite anpassen. Sie können dieses Verhalten akzeptieren, Sie können es sich explizit zu Nutze machen oder mit Hilfe verschiedener Techniken unterbinden (siehe beispielsweise Kapitel 5) - welchen Weg auch immer Sie wählen, kontrollieren Sie das Ergebnis auf jeden Fall in Ihrem Browser, das heißt, laden Sie die Webseite und schauen Sie, wie sich das Erscheinungsbild Ihrer Webseite verändert, wenn Sie das Browserfenster größer oder kleiner machen.



Auf diese Weise können Sie auch verschiedene Bildschirmauflösungen simulieren.



Wenn Sie eine Webseite im Browser begutachten, einen Fehler entdecken, diesen im HTML-Code beheben und dann in den Browser zurückkehren, wo Sie die Webseite neu laden, kann es sein, dass Ihre Korrekturen nicht sichtbar werden, weil der Browser die alte Webseite aus seinem Cache lädt. Um sicherzustellen, dass der Browser die Datei wirklich ganz neu lädt, müssen Sie beim Netscape-Browser die (Shift)-Taste gedrückt halten, während Sie den Befehl zum Neuladen (oder die entsprechende Schaltfläche) drücken, beim Internet Explorer müssen die (Strg)-Taste gedrückt halten, während Sie den Aktualisieren-Schalter drücken.

Doch nicht alle Elemente einer Webseite lassen sich auf diese Weise im Webbrowser überprüfen. Viele dynamische Elemente sind auf serverseitige Unterstützung angewiesen und können im Browser nur dann korrekt angezeigt werden, wenn Sie von einem Server geladen werden. Wenn Sie solche Elemente verwenden wollen, sollten Sie sich zuvor eine Testumgebung einrichten, unter der Sie diese Elemente kontrollieren und begutachten können. Dies führt uns zu Abschnitt 1.4 und der Einrichtung eines lokalen Webserver.

1.4 Testen auf einem lokalen Webserver

Wenn Sie einen eigenen Webserver betreiben, haben Sie natürlich alle Möglichkeiten, Ihre Webseiten und Webs, direkt auf dem Server zu erstellen und auszutesten - auch wenn diese Vorgehensweise aus sicherheitstechnischen Gründen nicht unbedingt zu empfehlen ist. Wenn Sie Ihre Webseiten auf einem privaten Rechner erstellen (oder allgemein irgendeinem Rechner, der nicht der Webserver ist, auf dem die Webseiten später veröffentlicht werden sollen), sollten Sie versuchen, auf diesem Rechner einen lokalen Webserver zu installieren. Zum einen ist die Einrichtung eines lokalen Webserver gar nicht sooo schwierig (auch wenn sie gelegentlich ihre Tücken hat), zum anderen können Sie Ihre Webseiten dann bequem und professionell testen.



Ein Webserver ist ein beliebiger Rechner, auf dem eine spezielle Webserver-Software läuft. Die Webserver-Software ist mit einem bestimmten Port verbunden (meist Port Nummer 80). Alle über diesen Port einkommenden HTTP-Anfragen (Anforderungen von Webseiten, Bilddateien, Applets oder Aufrufe von CGI-Programmen) werden von der Webserver-Software abgefangen und automatisch bearbeitet. Der Unterschied zwischen einem echten Webserver im Internet und einem lokalen Webserver ist letztlich nur der, dass der echte Webserver direkt mit dem Internet verbunden ist und eine entsprechende IP-Adresse besitzt. So können Internet-Teilnehmer aus der ganzen Welt auf ihn zugreifen und die von ihm verwalteten Webseiten anfordern. Der lokale Webserver ist dagegen üblicherweise nicht mit dem Internet verbunden, meist handelt es sich um einen alleinstehenden Rechner. Die Ausführung einer Webserver-Software auf einem solchen Rechner hat - abgesehen vom Austesten der eigenen Webseiten - wenig praktischen Nutzen (es können ja keine Anforderungen von anderen Rechnern eingehen). Als Mittelding zwischen dem echten Internet-Webserver und dem lokalen Webserver gibt es noch

Webserver, die in Intranets (lokalen Netzwerken von Firmen, Instituten, etc.) installiert sind, und dort zum Informationsaustausch zwischen den Teilnehmern genutzt werden.

Bezugsquellen für Webserver-Software

Für Webserver-Software muss man nicht unbedingt tief in die Tasche greifen. Bei den meisten Betriebssystemen gehört die Webserver-Software zur Software-Ausstattung - und wenn nicht, so gibt es genügend Webserver, die man sich kostenfrei aus dem Internet herunterladen kann.

- Den Apache-Server für Linux (oder auch Windows) können Sie beispielsweise von <http://www.apache.org> herunterladen. Wenn Sie Linux als Betriebssystem verwenden, dürfte er sogar mit ziemlicher Wahrscheinlichkeit bereits installiert oder zumindest auf Ihrer Linux-CD zu finden sein.
- Wenn Sie mit Windows arbeiten, werden Sie den PWS (Personal Web Server) oder den IIS (Internet Information Server) verwenden, die üblicherweise zusammen mit dem Betriebssystem ausgeliefert werden (Windows 98/2000. Ansonsten hilft das Option-Pack: <http://www.microsoft.com/msdownload/ntoptionpack/askwiz.asp> oder <http://www.microsoft.com/ntserver/nts/downloads/Recommended/NT4OptPk>). Sollten Sie mit den Windows-Produkten Schwierigkeiten haben, weichen Sie auf den Apache-Server für Windows (siehe oben) oder den OmniHttpd-Webserver von Omnicron (<http://www.omnicron.ab.ca>) aus.
- Für Windows und UNIX gleichermaßen geeignet ist der Xitami-Server, den man sich kostenfrei von der Website unter www.xitami.com herunterladen kann.



Ausführlichere Informationen zu Apache, PWS/IIS und OmniHTTPd finden Sie in Anhang A.

Das Installieren des Webserver

Wie Sie Ihren Webserver installieren müssen, hängt davon ab, für welche Webserver-Software Sie sich entschieden haben, welche Version der Server-Software Sie verwenden und mit welchem Betriebssystem Sie arbeiten. Zu viele Möglichkeiten, um sie alle im Rahmen dieses Buches zu beschreiben. Wir beschränken uns daher in diesem Abschnitt auf einige allgemeinere Anmerkungen, ausführlichere Informationen finden Sie im Anhang A und in der Originaldokumentation zu Ihrem Webserver.

Der Apache-Server unter Linux

Wenn Sie mit Linux arbeiten, prüfen Sie zuerst einmal, ob der Apache-Server nicht vielleicht schon installiert ist. Die Programmdatei des Apache-Servers heißt *httpd* und kann mit verschiedenen Optionen von der Kommandozeile aus aufgerufen werden. Um zu testen, ob der

Apache-Server installiert ist, verwenden wir die Option -v:

```
> httpd -v
```

Im Erfolgsfall wird Ihnen daraufhin die Versionsnummer des Servers angezeigt:

```
Server version: Apache/1.3.12 (unix) (SuSE/Linux)
Server built:   Jul 30 2000 22:47:29
```

Wenn Sie stattdessen eine Fehlermeldung erhalten, dass der Befehl nicht gefunden werden konnte, muss Sie das nicht gleich entmutigen - eventuell steht *httpd* nur nicht in Ihrem Pfad. Versuchen Sie es dann mit folgenden Aufrufen:

```
> /usr/sbin/httpd -v           // bevorzugtes Installationsverzeichnis
                               // für RedHat und SuSE
> /usr/locale/apache/httpd -v // für Original von www.apache.org
```

(Im Zweifelsfall halten Sie mit dem Shell-Befehl `find / -name httpd -print` nach der *httpd*-Datei Ausschau.¹)

Sollte der Apache-Server nicht installiert sein, schauen Sie nach, ob er zum Lieferumfang Ihrer Linux-Distribution gehört und sich auf den Linux-CDs befindet. Für die meisten Linux-Distributionen (beispielsweise RedHat, SuSE oder Mandrake) gehört der Apache-Server zum Standardumfang.

Schließlich besteht immer die Möglichkeit, sich die aktuelle Version von der originalen Apache-Website (<http://www.apache.org>) herunterzuladen. Sie müssen sich dann nur entscheiden, welche Version genau Sie verwenden wollen (auf der Website werden auch ältere Versionen angeboten) und ob Sie den Server als Binärdatei oder als Quelltextsammlung herunterladen wollen, und die Installationshinweise auf der Website lesen.

Wenn der Server nicht bereits automatisch beim Booten des Betriebssystems gestartet wird, können Sie ihn jederzeit manuell starten, indem Sie *httpd* ohne Option aufrufen:

```
> httpd
```

oder je nach Installation mit Pfadangabe:

```
> /usr/sbin/httpd
> /usr/locale/apache/httpd
```

Danach können Sie den Webserver testen. Rufen Sie Ihren Webbrowser auf und steuern Sie die Adresse `http://localhost` (oder `http://IhrRechnername`) an. Entweder erscheint jetzt die Standardwebseite Ihrer Apache-Installation (üblicherweise ein HTML-Dokument mit Informationen zum Webserver) oder Sie erhalten eine Meldung, dass der Webserver aus irgendeinem Grunde nicht kontaktiert werden konnte.

Der Apache-Server unter Windows

Mittlerweile gibt es auch eine Windows-Version des Apache-Servers. Das ist zweifelsohne eine ausgesprochen gute Nachricht, denn der Apache-Server ist sicherlich einer der bedeutendsten Webserver überhaupt, und die Portierung dieses kostenlosen, professionellen Tools auf die Windows-Plattform ist ohne Einschränkung zu begrüßen. Der einzige Wermutstropfen dabei ist, dass es augenblicklich (das heißt, zu der Zeit, da dieses Buch entstanden ist) noch keine vollkommen ausgereifte, stabile Version gibt. Vielleicht hat sich dies bis zum Erscheinen dieses Buches geändert, vielleicht auch nicht. Wie auch immer, wenn Sie Interesse an der Windows-Version des Apache-Servers haben, steuern Sie die Apache-Website an (<http://www.apache.org>), lesen Sie dort die aktuellen Informationen zur Windows-Version und laden Sie sich gegebenenfalls den Server und die Installationshinweise herunter.

Der PWS und der IIS

Die Standard-Webserver für die Windows-Plattformen, der Personal Webserver (PWS) und der Internet Information Server (IIS), werden zumeist direkt mit dem Betriebssystem ausgeliefert.

Welchen Server man nutzen kann und wie dieser zu installieren ist, hängt dabei von der Betriebssystem-Version ab: Windows 95 oder Windows 98, Windows NT 4.0 oder Windows 2000, Workstation- oder Server-Version. Je nach Betriebssystem wird der Webserver entweder zusammen mit dem Betriebssystem installiert, kann von der Windows-CD nachinstalliert werden oder ist im Option Pack enthalten, das wir zuletzt unter <http://www.microsoft.com/ntserver/nts/downloads/Recommended/NT4OptPk> beziehungsweise <http://www.microsoft.com/msdownload/ntoptionpack/askwiz.asp> gefunden haben (siehe auch Anhang A).



*Der Betrieb eines lokalen Webservers erfordert, dass das TCP/IP-Protokoll installiert ist. Für Anwender von Windows 95/98 oder Windows NT 4.0 Workstation bedeutet dies, dass Sie den Rechner gegebenenfalls eigenhändig als Client-Rechner konfigurieren müssen. Rufen Sie dazu **Start/Einstellungen/Systemsteuerung/Netzwerk** auf und fügen Sie - falls nicht schon geschehen - über den Schalter **Hinzufügen** die Netzwerkkomponenten **Client für Microsoft-Netzwerke und TCP/IP-Protokoll** hinzu.*

Der PWS und der IIS werden üblicherweise automatisch gestartet und erscheinen beim Programmstart als Symbol in der Taskleiste. Über das Kontextmenü des Symbols kann der Webserver angehalten und weiter ausgeführt werden kann.

- Unter Windows NT Workstation können Sie über Einstellungen/ Systemsteuerung/Dienste nachsehen, ob der Server läuft.
- Unter Windows 2000 Server können Sie unter (Achtung, der folgende Pfad ist etwas

länger) **Start/Programme/Verwaltung/Computerverwaltung**, Unterknoten **Dienste und Anwendungen/Internet-Informationdienste**, Eintrag **Standardwebsite** nachsehen, ob der Server läuft.

Zum Testen des Webservers rufen Sie Ihren Webbrowser auf und steuern die Adresse `http://localhost` (oder `http://IhrRechnername`) an. Entweder erscheint jetzt die Standardwebseite Ihres Servers (üblicherweise ein HTML-Dokument mit Informationen zum Webserver) oder Sie erhalten eine Meldung, dass der Webserver aus irgendeinem Grunde nicht kontaktiert werden konnte.

Der OmniHTTPd-Server

Nachdem Sie die EXE-Datei des OmniHTTPd-Webservers auf Ihre Festplatte heruntergeladen haben, brauchen Sie diese nur doppelt anzuklicken, um den Installationsvorgang zu starten. Folgen Sie den Anweisungen in den Dialogfeldern und richten Sie den Server so ein, dass er bei dem Booten automatisch geladen wird.

Läuft der Server erscheint in der Windows-Taskleiste ein entsprechendes Symbol für den Server. Läuft der Server nicht, kann er über seine Programmgruppe (Aufruf über **Start/Programme**) gestartet werden.

Zum Testen des Webservers rufen Sie Ihren Webbrowser auf und steuern Sie die Adresse `http://localhost` (oder `http://IhrRechnername`) an. Entweder erscheint jetzt die Standardwebseite Ihres Servers (ein HTML-Dokument mit Informationen zum Webserver) oder Sie erhalten eine Meldung, dass der Webserver aus irgendeinem Grunde nicht kontaktiert werden konnte.



Sie können keine zwei Webserver gleichzeitig ausführen, wenn diese den gleichen Port (für Webseiten üblicherweise 80) überwachen. Wenn Sie also beispielsweise den OmniHTTPd-Server verwenden wollen, achten Sie darauf, dass Sie nicht bereits zuvor schon einmal den PWS oder IIS installiert haben und dieser ebenfalls automatisch beim Booten gestartet wird.

Das Einrichten von Webs/Webseiten unter dem Webserver

Der nächste Schritt ist, dass wir unsere Testdatei auf den Webserver »hochladen« und mit dem Browser vom Webserver abrufen (das heißt, wir geben im Adressfeld des Browsers nicht mehr den Dateipfad zu der Webseite, sondern ihren URL an).

Das »Hochladen« auf unseren lokalen Webserver gestaltet sich denkbar einfach. Wir müssen die Datei nur in eines der vom Webserver verwalteten Verzeichnisse zu kopieren. Wo aber liegen diese Verzeichnisse?

Bei der Installation des Webservers wurde für diesen auf Ihrer Festplatte ein eigenes Verzeichnis eingerichtet und fest mit dem Servernamen verbunden. Dieses Verzeichnis ist das Wurzelverzeichnis für alle Webseiten und Webs, die auf dem Webserver veröffentlicht werden - das sogenannte Dokumentenverzeichnis. Anders ausgedrückt, wenn Sie eine Webseite auf Ihren lokalen Webserver »hochladen« wollen, kopieren Sie die Seite einfach in das Dokumentenverzeichnis. (Oder organisieren Sie Ihre Webseiten, indem Sie unter dem Dokumentenverzeichnis Unterverzeichnisse anlegen, auf die Sie Ihre Webseiten verteilen. Üblicherweise legt man für jedes Web ein eigenes Unterverzeichnis an, das je nach Strukturierung des Webs selbstverständlich auch wieder Unterverzeichnisse enthalten kann.) Jetzt müssen Sie nur noch in Erfahrung bringen, wie das Dokumentenverzeichnis Ihres lokalen Webservers heißt.



Die in diesem Abschnitt angegebenen Pfadangaben sind Standardwerte, die bei der Installation verändert werden können. Es ist also durchaus möglich, dass auf Ihrem Rechner abweichende Pfade und Verzeichnisse verwendet werden.

- Die Konfiguration des Apache-Servers erfolgt größtenteils über die Datei **httpd.conf**. Diese Datei befindet sich je nach Version in einem der Verzeichnisse */etc/httpd/*, */etc/httpd/conf* oder */usr/local/apache/conf*. Suchen Sie in der Datei nach einem Eintrag für DocumentRoot. Gibt es keinen entsprechenden Eintrag, verwendet der Server höchstwahrscheinlich */home/httpd/html*, */usr/local/httpd/htdocs* oder */usr/local/apache/htdocs* als Dokumentenverzeichnis.
- Das Dokumentenverzeichnis des Personal Web Servers (wie auch des IIS- Webservers) lautet standardmäßig *C:/inetpub/wwwroot*. Sie können entweder auf Ihrer Festplatte nachsehen, ob ein entsprechendes Verzeichnis angelegt wurde oder das Verwaltungsprogramm zu Ihrem Webserver aufrufen. Im Falle des Personal Web-Managers brauchen Sie dazu nur auf das Symbol des Webservers in der Taskleiste doppelzuklicken. Wenn Sie mit dem IIS-Webserver arbeiten, müssen Sie das Administrationstool für die Internet-Informationdienste aufrufen (für Windows 2000 Server erfolgt der Aufruf über **Start/Programme/Verwaltung/ Computerverwaltung**, Unterknoten **Dienste und Anwendungen/Internet- Informationsdienste**, Befehl **Eigenschaften** im Kontextmenü zu **Standardwebsite**, Registerkarte **Basisverzeichnis**).
- Das Dokumentenverzeichnis des OmniHttpd lautet **C:/httpd/HtDocs**, sofern Sie den Server in das Verzeichnis **C:/httpd** installiert haben.

Speichern oder kopieren Sie jetzt die in Abschnitt 1.2 erstellte Test-Webseite in das Dokumentenverzeichnis Ihres Servers (unter Linux benötigen Sie dazu entsprechende Schreibrechte, also eventuell als root einloggen) und laden Sie sie über einen Webbrowser (Internet Explorer oder Netscape Navigator). Geben Sie dabei als Adresse (URL) nicht das Dokumentenverzeichnis an, sondern den Namen des Webservers, gefolgt vom Namen der Datei.

Wenn Sie die Webseite *testseite.html* genannt und direkt im Dokumentenverzeichnis Ihres lokalen Webservers gespeichert haben, verwenden Sie als URL: *http://localhost/testseite.html*.

Wenn Sie unter dem Dokumentenverzeichnis des Webservers extra ein Unterverzeichnis *test* angelegt und die Test-Webseite in diesem abgespeichert haben, schreiben Sie: *http://localhost/test/testseite.html*.

Wenn Ihr lokaler Rechner einen eigenen Namen hat, können Sie auch diesen verwenden: *http://localhost/testseite.html* oder *http://localhost/test/testseite.html*.

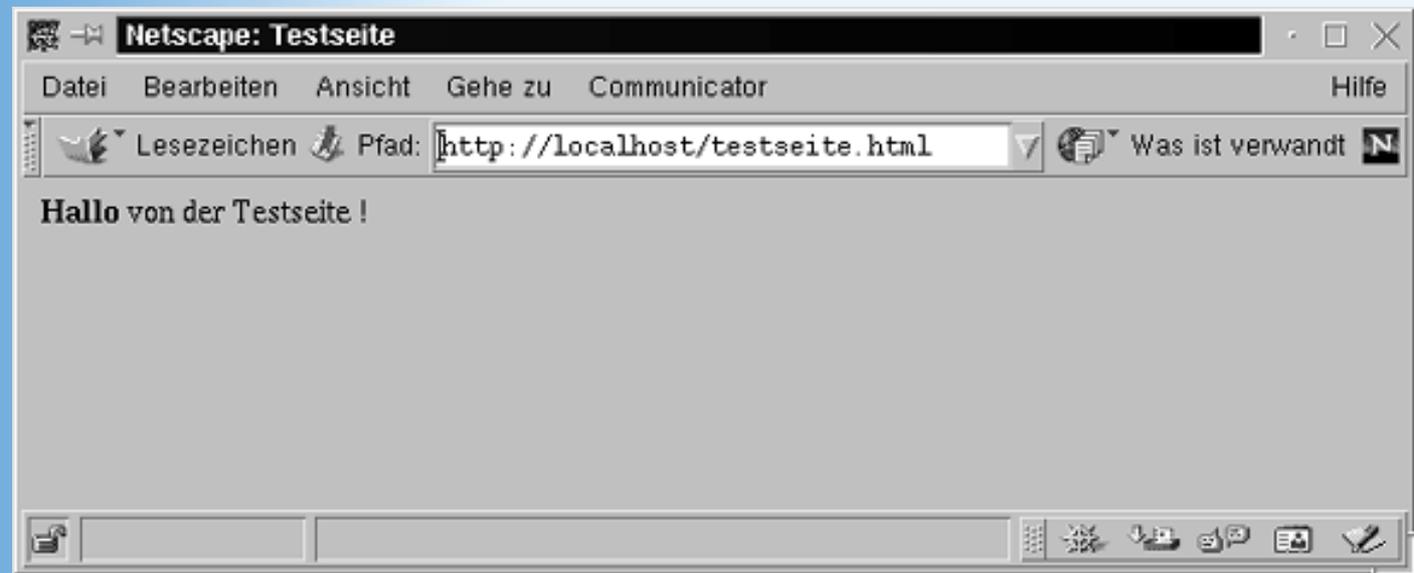


Abbildung 1.4: Anfordern der Testseite vom lokalen Webserver

1.5 Der Schritt in die Öffentlichkeit - das Hochladen auf den Internet-Webserver

Hat man die eigene Webseite (oder das eigene Web) fertig gestellt und auf Herz und Nieren geprüft, besteht der letzte Schritt darin, die Webseite (das Web) zu veröffentlichen, indem man es auf einen Internet-Webserver hochlädt. Wenn Sie selbst einen Webserver im Internet betreiben, werden Sie damit wohl keine Probleme haben. Glücklicherweise auch, wer in einer Firma arbeitet und seine Webseiten über den Webserver des firmeneigenen Intranets verbreiten will, er oder sie muss sich nur vertrauensvoll an den zuständigen System- oder Serveradministrator wenden (Pech allerdings, wenn sich in Ihrer Firma für solche Aufgaben niemand zuständig fühlt. Gehen Sie in einem solchen Fall unbedingt mit Diskretion und Geschick an die Sache, sonst spricht sich schnell herum, dass Sie sich ja mit der Verwaltung des Webservers bereits bestens auskennen, und Sie werden, ohne dass Sie sich wehren können, zum ehrenamtlichen Serveradministrator ernannt)

Schließlich bleibt die Möglichkeit, dass Sie Kunde bei einem kommerziellen Internet-Provider sind, und Ihre Webseiten über dessen Webserver veröffentlichen (in einem eigenen Unterverzeichnis des Servers oder auch unter einem eigenen Domännennamen, siehe nächster Abschnitt). Dazu benötigen Sie

- Ein FTP-Programm zum Hochladen der Webseiten (und aller zugehörigen Dateien,

beispielsweise Bilddateien, Applets, externe Stylesheets, etc.) - beispielsweise der Webpublishing-Assistent von Microsoft oder FTS_PRO (aktuelle Demoversion kann von <http://www.ipswitch.com/> heruntergeladen werden, eine Shareware-Version zur privaten Nutzung kann man über http://www2.bibl.fh-koeln.de/elektra/ftp_hilfe.html beziehen).

- Informationen über den FTP-Server, auf den Sie die Dateien hochladen, das Zielverzeichnis für die Dateien sowie Name und Kennwort zum Einloggen auf den Server. Wenden Sie sich hierzu an Ihren Internet-Provider.
- Informationen über die Konfiguration Ihres Webs (wie viele MBytes stehen zur Verfügung, kann man Unterverzeichnisse einrichten, auf welche Extension müssen die HTML-Dateien auslauten, wie muss die Startseite des Webs heißen). Wenden Sie sich hierzu an Ihren Internet-Provider.



*Wenn Sie Mitglied bei T-Online, CompuServe oder einem anderen großen Internet-Provider sind, halten Sie am besten zuerst einmal auf der Website Ihres Providers nach Unterstützung und Informationen zum Thema »Private Homepages« Ausschau. Für T-Online derzeit unter **Service/Web-Präsenz/mehr Info** (zu Private Homepage bei T-Online) = <http://www.t-online.de/service/index/wphsvxaa.htm>; für CompuServe derzeit unter **Mitglieder/Ourworld/Eigene Homepage** = <http://www.compuserve.de/mitglieder/ourworld/eigenehomepage/start.jsp>; für AOL derzeit unter Keyword HOMETOWN = <http://hometown.aol.de>*

Anhand von WS_FTP (<http://www.ipswitch.com/>) möchte ich Ihnen kurz die Dateiübertragung mit einem FTP-Programm vorstellen.

Nach dem Starten von WS_FTP sehen Sie ein zweigeteiltes Fenster. Links werden die Verzeichnisse Ihres lokalen Systems angezeigt, rechts die Verzeichnisse des entfernten Webservers (sobald wir uns auf diesem eingeloggt haben).

1. Klicken Sie im linken Teil auf den Schalter **ChgDir** und wechseln Sie zu dem Wurzelverzeichnis des hochzuladenden Webs (bzw. des Verzeichnisses, in dem die hochzuladenden Dateien stehen).

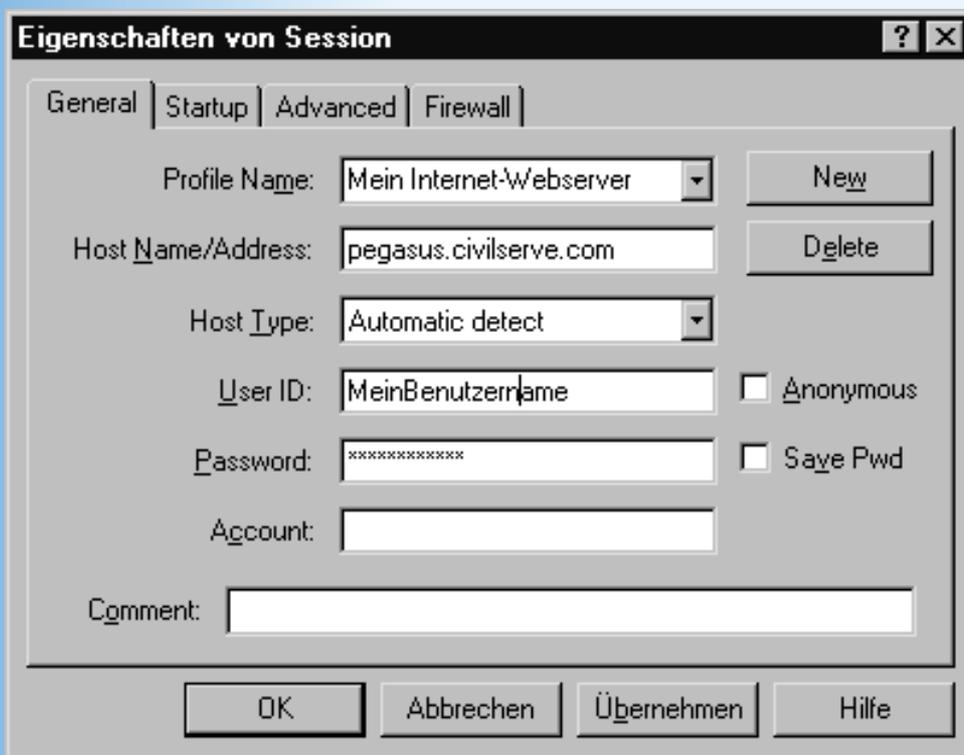


Abbildung 1.5: Verbindungsdaten eingeben

2. Klicken Sie auf den Schalter **Connect**. In dem erscheinenden Dialogfenster geben Sie auf der Registerkarte **General** einen frei wählbaren Namen für die herzustellende Verbindung (Profil), die Adresse des FTP-Servers, auf den Sie die Daten hochladen wollen, sowie den Benutzernamen und das Passwort, die Sie von Ihrem Webserver- Provider zugeteilt bekommen haben, ein. Drücken Sie noch nicht auf **OK**!
3. Stellen Sie jetzt eine Internetverbindung her (Anwahl über Modem, etc.)
4. Drücken Sie jetzt im Dialogfenster von WS_FTP auf **OK**, um sich auf dem Rechner Ihres Webserver-Providers einzuloggen.

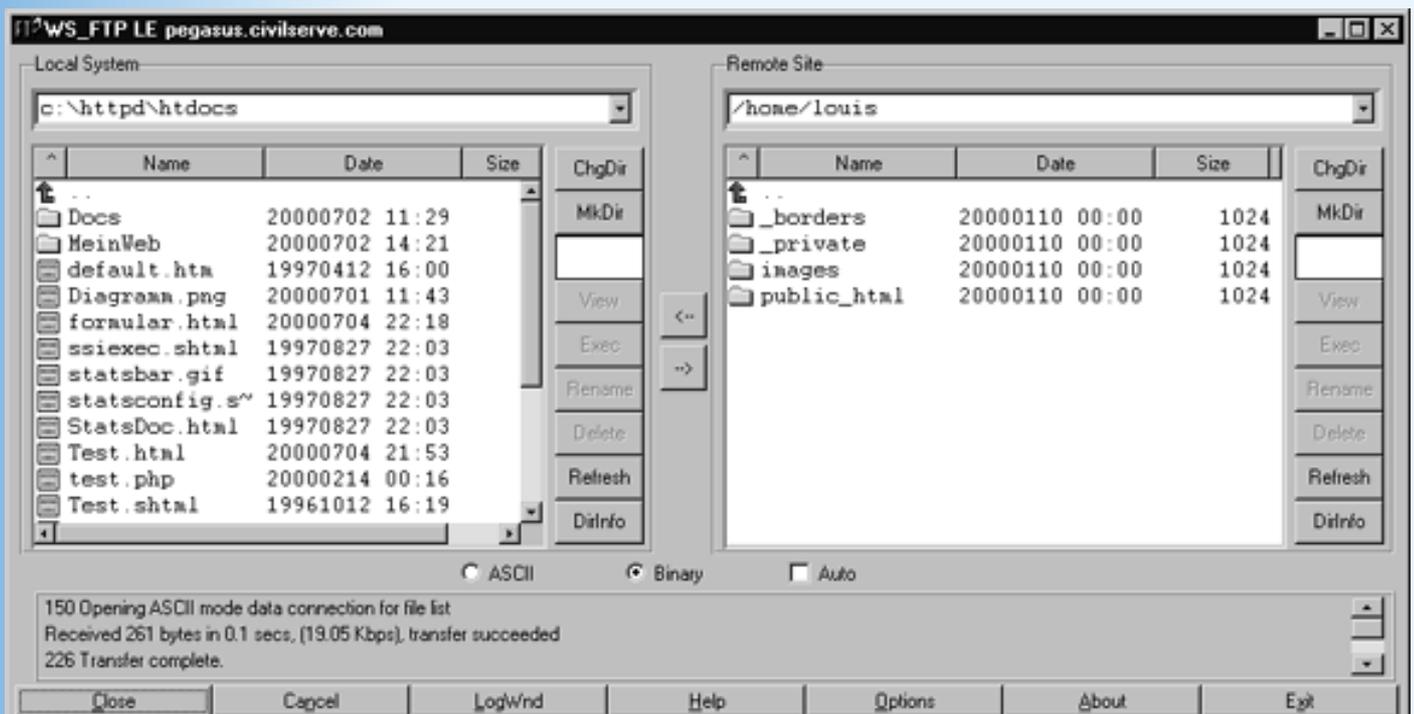


Abbildung 1.6: Die Verbindung zum Webserver wurde hergestellt

5. Klicken Sie im rechten Teil auf den Schalter **ChgDir** und wechseln Sie zu dem Verzeichnis in das Sie Ihre Webdateien hochladen wollen (unter Umständen beim Webserver-Provider zu erfragen).
6. Markieren Sie im linken Fenster die Dateien und Unterverzeichnisse, die Sie hochladen wollen (Mehrfachauswahl ist bei gedrückter (Strg)- oder (Shift)-Taste möglich).
7. Drücken Sie auf den Pfeil-Schalter von WS_FTP, um die Dateien auf den Webserver zu übertragen.
8. Beenden Sie die Verbindung.

1.6 Die richtige Wahl des Internet Service Providers

Um Ihre Webseiten im Internet zu publizieren, müssen Sie die Dateien auf einen Rechner kopieren, auf dem sie rund um die Uhr den ganzen Tag erreichbar sind. Diese Dienstleistung und den nötigen Festplattenplatz bieten so genannte Internet Service Provider an.

Die Qual der Wahl

Wie bei allen Sachen dieser Welt gibt es auch bei den Service Providern qualitative Unterschiede. Das beginnt bei Providern, die den gesamten Service für Sie kostenlos anbieten. Die Rechnung zahlt dann der Besucher, der Ihre Website ansteuert und dort mit fremder Werbung berieselt wird. Das ist bei einer privaten Homepage für den Besucher schon eine Zumutung, für einen professionellen Unternehmensauftritt aber völlig ungeeignet. Je mehr Service und Dienstleistungen (wie etwa eine Datenbank) benötigt werden, um so höher wird der meist monatlich zu bezahlende Preis. Preisvergleiche lohnen sich, aber denken Sie daran, dass es Qualität nicht umsonst gibt.



Achten Sie auch darauf, dass der Provider Ihnen eine Zugriffsstatistik zur Verfügung stellt, aus der Sie entnehmen können, wie oft Ihre Seiten aufgerufen werden.

Vom Unterverzeichnis zur eigenen Domain - der Name Ihrer Website

Die Provider stellen natürlich nicht für jeden Kunden einen eigenen Rechner auf. Dann wäre das Angebot viel zu teuer. Ihre Website wird vielmehr als Unterverzeichnis oder virtueller Webserver auf einem Rechner installiert, wo noch viele andere Kunden ihre Daten und HTML-Seiten ablegen. Erst dadurch kommt ein recht geringer Preis für diese Dienstleistung zustande. (Um so mehr Kunden ein Provider hat, um so günstiger ist meist der Preis; aber um so mehr Kunden teilen Sie sich dann auch die Leistung des Rechners, der diese Websites abarbeiten muss.)

Wie können Websurfer Ihre Webseiten aufrufen?

Damit Ihre Webseiten im Internet aufgerufen werden können, müssen sie über einen

eindeutigen Namen verfügen. Dieser Name, der weltweit einmalig ist und auch URL genannt wird, gliedert sich in zwei Bestandteile. Als Beispiel soll uns die fiktive Adresse: *www.marktundtechnik.de/user/index.html* dienen.

- Der erste Bestandteil ist der Name des Rechners, auf dem Ihre Seiten liegen (*www.marktundtechnik.de*).
- Hinter dem / kann der Name eines oder mehrerer Verzeichnisse stehen (*/user/*), gefolgt vom Namen der eigentlichen HTML-Datei (*index.html*).



*Wird nur ein Rechnernamen, gegebenenfalls mit Verzeichnispfad aber ohne Dateiname angegeben, sucht der Webserver in dem betreffenden Verzeichnis nach einer Datei, die einen für den Server spezifischen Dateinamen trägt - meist *index.html* oder *index.htm*. Deshalb reicht auch die Angabe *http://www.marktundtechnik.de/user/*, um die Seite *index.html* aus dem Verzeichnis *http://www.marktundtechnik.de/user/* aufzurufen. Erkundigen Sie sich bei Ihrem Service Provider nach dem oder den Standardnamen des Servers und speichern Sie Ihre Eintritts- oder zentrale Webseite unter diesem Namen.*

Die einfachste und meist auch kostenlose Möglichkeit ist, bei einem Provider unter einem eigenen Verzeichnis gefunden zu werden. Das ist zum Beispiel bei T-Online möglich. Dort lautet der Name Ihrer Website dann: *home.t-online.de/home/username/* - sofern der Name Ihrer Startseite *index.html* ist. (Ansonsten muss man den Namen explizit im URL mit angeben.)

Die nächste Stufe ist ein eigener Domainname unterhalb (von rechts nach links gelesen) des Namen des Providers. Dann würde Ihre Website zum Beispiel unter der fiktiven Adresse: *www.ihrname.providernamen.de* gefunden werden.



Die meisten Webserver sind heutzutage in der Lage, sogenannte virtuelle Domains bereitzustellen. Dazu werden der Webserver und der Rechner so konfiguriert, dass Anfragen zu den verschiedenen Domains auf jeweils eigene Verzeichnispfade abgebildet werden. Das ist für Sie als Anwender normalerweise völlig transparent. Sie sind dann einer von vielen Kunden eines Providers, die auf einem Rechner ihre Dateien ablegen.

Für professionelle Anwender ist ein markanter eigener Name für das Webangebot unerlässlich. Zum einen aus Imagegründen, zum anderen, um zu gewährleisten, dass Interessenten und Kunden die Website der Firma über den Firmennamen finden können (wenn die Website unter

dem Namen *www.firmenname* abgelegt ist). Ein weiterer Vorteil ist, dass man im Falle eines eigenen Domainnamens bei Bedarf den Provider wechseln kann, ohne dass es die Besucher der Website bemerken.

Rechtliche Aspekte bei Wahl eines Domainnamens

Mit den eigenen Domainnamen sollten wir uns noch ein wenig intensiver beschäftigen.

Der Name untergliedert sich in eine Top-Level-Domain (.de) und eine Second-Level-Domain (firmenname).

Die Top-Level-Domain .de ist das internationale Kürzel im Internet für Deutschland. Jedes Land dieser Erde hat ein eigenes zweibuchstabiges Kürzel. In jedem Land werden die darunter liegenden Second-Level-Domainnamen (von rechts nach links betrachtet) selbstständig verwaltet. In Deutschland ist das DeNIC (Deutsches Network Information Center, *www.denic.de*) in Frankfurt dafür zuständig, eine bundesweite Datenbank mit diesen Informationen zu betreiben. Da eine Internetadresse eindeutig sein muss, kann also ein Name unterhalb von .de nur ein einziges Mal vergeben werden. Da immer mehr auch finanzielle Interessen hinter einem Webangebot stehen, sind attraktive Adressen sehr begehrt. Deshalb ist die Namensvergabe im Internet immer häufiger auch Streitpunkt vor Gericht. In den letzten Jahren war die Rechtssprechung in diesem Bereich recht uneinheitlich, da sich auch Richter und Anwälte in diesem neuen Bereich erst einarbeiten mussten und noch keine Präzedenzurteile vorlagen. Aufgrund der sehr langen Dauer von Gerichtsverfahren in Deutschland liegen erst seit kurzer Zeit Gerichtsurteile vor, die auch Rechtsgültigkeit haben. Viele Entscheidungen wurden zuvor vorläufig per einstweiliger Verfügung entschieden.

Auf jeden Fall kommen die markenrechtlichen Regeln bei den richterlichen Entscheidungen zur Anwendung. Wenn Sie sich also einen Namen für Ihre Website ausdenken, müssen Sie darauf achten, nicht die Rechte Dritter zu verletzen. Sonst droht Ihnen irgendwann eine einstweilige Verfügung, womit Sie den von Ihnen gewählten Namen wieder abgeben müssen. Und wenn Sie dann schon viel Geld in Visitenkarte und Briefkopf mit Ihrem Webseitenamen investiert haben, ist das sehr ärgerlich. Das kann Ihnen übrigens auch dann passieren, wenn der Eintrag auf Markenschutz erst nach Ihrer Wahl des Domainnamens und Betrieb Ihrer Website stattfindet. In dieser Hinsicht gilt bei den deutschen Richtern kein: »Wer zuerst kommt, mahlt zuerst.«

Es gibt auch kuriose Fälle, wo geklagt wird, weil sich ein Domainname nur mit einem Buchstaben von seinem eigenen, geschützten Namen unterscheidet. Der Kläger befürchtete eine Verwechslungsgefahr. Allerdings sind solche Dinge momentan noch nicht letztinstanzlich geklärt.

Schließlich hat man die Möglichkeit, sich mit jemandem, der unter gleichem Namen erreichbar sein möchte, zu einigen. Dann wird eine gemeinsame Startseite ins Netz gestellt, von wo aus der Besucher auf das jeweilige Angebot verwiesen wird (siehe zum Beispiel <http://www.winterthur.ch>).



Bei der Wahl des Namens müssen Sie beachten, dass Sie keine deutschen Umlaute verwenden dürfen und als einziges Sonderzeichen der Bindestrich - zulässig ist. Auch Leerzeichen sind nicht erlaubt (www.bär gmbh.de ist also nicht möglich). Die Länge des Namens darf maximal 256 Buchstaben betragen.

Wenn Sie dies alles geklärt haben, können Sie sich entweder persönlich an einen Provider bei Ihnen vor Ort wenden und ein Angebot machen lassen, wie hoch bei ihm einmalige Einrichtungskosten und laufende monatliche Kosten für den Betrieb Ihres virtuellen Webservers sind; oder Sie wenden sich an die bundesweit agierenden Provider, die meist nur über das Internet erreichbar sind.

Der Provider sorgt dann auch dafür, dass Ihr gewünschter Domainname in die Datenbank des DeNIC eingetragen wird. Darum müssen Sie sich also selbst nicht kümmern.

Bei größeren professionellen Auftritten sollte man schon bei Beginn der Planung das Gespräch mit einem Provider suchen. Dieser kann Ihnen dann Tipps und Hinweise geben, was bei der Umsetzung eines Internetauftritts vom Papier in den Rechner alles beachtet werden muss.

1.7 Zusammenfassung

Heute war ein Tag zum Kennenlernen. Wir haben ein wenig über Webdesign geplaudert und uns die verschiedenen Werkzeuge angeschaut, die wir für die Erstellung von Webseiten und Webs benötigen:

- den Editor zum Aufsetzen des HTML-Codes
- den Browser zum Anschauen und Kontrollieren der Webseiten
- den lokalen Webserver zum Testen (insbesondere serverseitiger, dynamischer Webinhalte)
- das FTP-Tool zum Hochladen des Webs auf den Internet-Webserver

Ausgelassen haben wir lediglich die Beschreibung spezieller Software zur Erstellung multimedialer Inhalte (Hintergründe, Grafiken, Diagramme, Videos, Sounddateien, etc.). In den nachfolgenden Kapiteln werden wir noch Gelegenheit haben, auch auf diese Werkzeuge ein wenig einzugehen.

1.8 Fragen und Antworten

Frage:

Kann ich meine Webseiten auch in Word aufsetzen?

Antwort:

Ja und nein! Was natürlich nicht geht, ist den Text einer Webseite in Word aufzusetzen, das

Ergebnis als DOC-Datei zu speichern und dann im Windows Explorer die Dateiendung in .html zu ändern und zu glauben, man hätte auf diese Weise eine HTML-Webseite gezaubert. (Laden Sie das Ergebnis doch mal in den Notepad-Editor und schauen Sie sich an, wie eine solche Seite aus Sicht des Webbrowsers aussieht!)

Antwort:

*Was Sie machen können, ist den in Word aufgesetzten Text explizit als ASCII- Text abzuspeichern (Dateityp »**Nur Text (*.txt)**« im **Speichern unter**- Dialog). Allerdings gehen dabei etliche Formatierungen (unter anderem Listen und Tabellen) verloren. Eine bessere Alternative ist es, den Text als HTML-Datei abzuspeichern. Word 2000 ist darin ziemlich perfekt, so dass die Word- Formatierungen nahezu 1:1 in HTML konvertiert werden (Menübefehl **Datei/ Als Webseite speichern**).*

Frage:

Soll ich mir einen professionellen Webeditor kaufen? Wenn ja, welchen?

Antwort:

Tja, da ist nur schwer zu raten. Um mit diesem Buch arbeiten und ansprechende Webseiten erstellen zu können, brauchen Sie jedenfalls keine aufwendige und teure Software. Ein einfacher Texteditor, ein kostenlos verfügbarer Webserver, FTP-Software zum Hochladen Ihres Webs und ein leistungsfähiges Shareware- Programm zur Erstellung und Aufbereitung von Grafiken (beispielsweise Paint Shop Pro, das der Buch-CD beiliegt) reichen vollkommen aus.

Antwort:

Wer professionell in die Erstellung von Webseiten einsteigen will, der sollte sich nach einem speziellen HTML-Editor und geeigneter Grafiksoftware umsehen.

Antwort:

Für den semiprofessionellen Bereich ist beispielsweise Microsoft FrontPage interessant. Es verfügt über einen sehr guten HTML-Editor und einer integrierter Webseitenverwaltung. Daneben gibt es viele vorgefertigte Design-Elemente, die vor allem für weniger erfahrene Webdesigner ohne Programmierkenntnisse interessant sind. Je nach Edition wird Microsoft FrontPage zusammen mit der Bildbearbeitungssoftware Image Composer vertrieben.

Antwort:

Ebenfalls sehr beliebt ist die DreamWeaver-Software von Macromedia (www.macromedia.com)

Antwort:

Für den High-End-Bereich könnte man sich auch eine Lösung aus kostengünstigem, weil kostenlosem, HTML-Editor (beispielsweise der HTML-Kit, den man von www.chami.com/html-kit herunterladen kann) und einem professionellen Grafikprogramm wie dem Adobe Photoshop vorstellen.

Frage:

Ich habe Probleme mit der Installation eines lokalen Webserver. Wo finde ich Hilfe?

Antwort:

Wenn Ihnen die Ausführungen in diesem Kapitel nicht genügen, schauen Sie doch einmal in Anhang A hinein. Sollten Sie auch dort keine Hinweise finden, die Ihnen weiterhelfen, schlagen Sie in der Dokumentation zu der Webserver- Software nach oder wenden Sie sich an den Hersteller der Webserver-Software (im Zweifelsfall immer erst mal im Internet auf den Websites der Firmen nachschauen).

1.9 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

Quiz

1. Was hatte Shakespeare im Sinn als er Polonius zu seinem nach Frankreich reisenden Sohn Laertes sagen ließ:

Die Kleidung kostbar, wie ´s dein Beutel kann,

Doch nicht ins Grillenhafte; reich , nicht bunt:

Denn es verkündet oft die Tracht den Mann,

Und die vom ersten Rang und Stand

Sind darin ausgesucht und edler Sitte.

2. Warum sollte man seine Webseiten stets in mehreren Webbrowsern austesten?

Übungen

Heute gibt es im Grunde nichts zum Üben. Aber vielleicht waren Sie beim Durcharbeiten dieses Kapitels ein wenig träge oder saßen gerade nicht an Ihrem Rechner. Dann wäre jetzt vielleicht eine gute Gelegenheit, die folgenden Übungen nachzuholen.

1. Installieren Sie auf Ihrem System einen lokalen Webserver (sofern nicht schon vorhanden).
 2. Probieren Sie, eine selbst erstellte Test-Webseite auf Ihrem lokalen Webserver zu installieren und mit einem Browser von dort zu laden.
 3. Falls Sie noch nie Webseiten im Internet veröffentlicht haben, setzen Sie sich mit Ihrem Internet Provider in Verbindung, klären Sie, wie Sie Webseiten auf seinen Webserver hochladen können und versuchen Sie, Ihre erste Webseite ins Internet zu stellen.
-

Die Konfigurationsdateien für den Apache sollten Sie sich ruhig einmal durchlesen - auch um sich ein Bild von der Leistungsfähigkeit dieses WWW-Servers machen zu können. Die Dateien finden Sie standardmäßig unter */etc/httpd* oder unter */usr/local/apache/conf*. Die wichtigste Datei ist dabei die Datei *httpd.conf*.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

Tag 2

Am Anfang war... HTML

HTML steht für HyperText Markup Language (Hypertext-Auszeichnungssprache) und ist bekanntlich die Muttersprache aller Webseiten. Bevor wir uns mit dynamischem HTML, CSS, Scripting und serverseitigen Technologien auseinandersetzen, sollten wir uns daher auf die Wurzeln all dieser wunderbaren, fortgeschrittenen Techniken besinnen. Da HTML im Grund eigentlich ganz einfach ist (und vielen Lesern wohl auch schon bekannt sein wird), werden wir uns mit diesem Grundkurs nicht allzu lange aufhalten. Wir werden die Konzepte und Ideen hinter HTML herausarbeiten, die wichtigsten HTML-Elemente in angemessener Breite vorstellen (so dass Sie dieses und das nachfolgende Kapitel auch zum Nachschlagen verwenden können) und - wo sinnvoll - ein wenig tiefer in die Praxis des Webdesigns mit reinem HTML abtauchen. Für alte HTML-Hasen also nicht viel Neues, für Anfänger aber eine ganze Menge Stoff.

Die Themen heute:

- Was ist eigentlich eine Markup-Sprache?
- Das Grundgerüst einer Webseite
- Seitengestaltung mit HTML-Tags
- Wie erstellt man Listen?
- Wie bindet man Bilder ein?
- Wie werden Bilder für die Veröffentlichung im Web aufbereitet
- Wie richtet man Hyperlinks ein?
- Auszeichnung von Blockgruppen und Textpassagen mit `<div>` und ``
- Moderne Formatierung mit Inline-Stilen
- Hintergrundbilder

2.1 Was sind Markups?

Ich hoffe, Sie sitzen noch an Ihrem Schreibtisch und nicht bereits im Bett, mit einem Glas warmer Milch als Nachtrunk neben sich und diesem Buch als Abendlektüre auf den Knien. Alle Leser, die mit dem Begriff »Markup« oder »Auszeichnung« nichts Richtiges anfangen können, obwohl Sie es schon tausendmal gehört haben, möchten wir jetzt nämlich bitten, einen Stift und einen Schreibblock zur Hand zu nehmen, und folgenden Text aufzusetzen (siehe Abbildung 2.1).

Die Kiefer in der chinesischen Malerei

In der traditionellen chinesischen Malerei gehört die Kiefer zu den am häufigsten dargestellten Bäumen überhaupt. Diese Vorrangstellung hat sie ihrer Gewässrauhheit und ihren immergrünen Nadeln zu verdanken, die sie in der chinesischen Vorstellung zu einem Symbol für langes Leben und Beständigkeit werden ließen. So gehört die Kiefer auch zu den drei Freunden im Winter, den langes Leben symbolisierenden Pflanzen Kiefer, Bambus und Pfauweide.

Abbildung 2.1: Zu formatierender Text

Stellen Sie sich vor, dieser Text soll Aufnahme in eine Broschüre über »Traditionelle chinesische Malerei« finden und Sie müssen den Text morgen beim Setzer abliefern. In der Broschüre soll die Überschrift durch eine fette, größere Schrift hervorgehoben und vom nachfolgenden Text etwas abgesetzt werden. Zudem soll die Textpassage »drei Freunde im Winter« kursiv abgedruckt werden. Wie teilen Sie dies dem Setzer mit?

Ganz einfach! Sie vermerken die Formatierungsbefehle auf dem Blatt (siehe Abbildung 2.2).

Die Kiefer in der chinesischen Malerei

fett
größere Schrift

In der traditionellen chinesischen Malerei gehört die Kiefer zu den am häufigsten dargestellten Bäumen überhaupt. Diese Vorrangstellung hat sie ihrer Gewässrauhheit und ihren immergrünen Nadeln zu verdanken, die sie in der chinesischen Vorstellung zu einem Symbol für langes Leben und Beständigkeit werden ließen. So gehört die Kiefer auch zu den drei Freunden im Winter, den langes Leben symbolisierenden Pflanzen Kiefer, Bambus und Pfauweide.

kursiv

Absatz linksbündig

Abbildung 2.2: Text mit Formatierungsanweisungen



Die Vermerke zur Formatierung des Textes sind die Markups oder Auszeichnungen, mit denen wir uns in diesem Abschnitt beschäftigen wollen.

Stellen Sie sich jetzt vor, Sie wollten den Text als Webseite im Internet veröffentlichen. Abgesehen davon, dass Sie den Text nun am Computer aufsetzen müssen, stellt sich die Frage, in welcher Form Sie nun die Markups eingeben sollen? Eine Frage, die direkt zu einer anderen Frage führt: Wer ist unser Setzer?

Das Setzen der Webseiten übernehmen die Webbrowser. Voraussetzung ist allerdings, dass der Webbrowser die Formatierungsanweisungen, die Markups, versteht. Dies gilt für den Webbrowser eines deutschen Besuchers ebenso wie für den Browser eines Chinesen, für den Browser eines Linux-Anwenders ebenso wie für den Browser eines Mac-Fans, für den Netscape-Browser ebenso wie für den Internet Explorer, Lynx oder Mosaic. Es dürfte klar sein, dass wir hier mit irgendwelchen frei formulierten Markups nicht mehr weiter kommen. Was wir brauchen, ist ein übergreifender Standard, der einen Satz fester Markups und Regeln zu deren Verwendung definiert. Dieser Standard ist HTML.

Was macht der Webbrowser aus den HTML-Markups?

Eigentlich, so muss man mit einem lachendem und einem weinenden Auge feststellen, eigentlich ist HTML zur Gestaltung von anspruchsvollen Webseiten absolut ungeeignet. Wir dürfen dies HTML jedoch nicht anlasten, schließlich war es nie dazu gedacht gewesen, die Ansprüche ambitionierter Internet-Fans und absatzorientierter Firmen zu erfüllen.

Als Tim Berners-Lee 1989 am CERN-Institut in Genf, dem Europäischen Institut für Teilchenphysik, einen Vorschlag zur Veröffentlichung von Hypertext-Dokumenten über das Internet unterbreitete, ahnte er sicherlich nicht, welche enorme Popularität und Verbreitung sein Projekt finden würde. Ihm ging es vor allem darum, einen Weg zu finden, wie man wissenschaftliche Dokumente und Informationen Plattform unabhängig, schnell und bequem über das Internet austauschen konnte. Dazu gehörte auch, dass sich die Forscher, die Ihre Dokumente und Daten im Internet veröffentlichen wollten, sich nicht unnötig mit komplizierten Formatierungsbefehlen herumplagen mussten. Zusammen mit Anders Berglund, der ihm bei der Ausarbeitung der ersten HTML-Spezifikation zur Seite stand, löste Berners-Lee dieses Problem, indem er nur wenige physikalische Formatierungsanweisungen (wie Fettschrift oder Kursivschrift) aufnahm, dafür aber Markups zur Kennzeichnung typischer Dokumentelemente wie Überschriften, Listen oder Hyperlinks einführte. Letztere beruhen auf der Idee, dass man die genaue Formatierung von Überschriften, Listen, Links, etc. getrost ganz dem Webbrowser überlassen kann. Der Webautor muss durch die Markups also nur noch anzeigen, welche Textstellen als Überschriften oder Listen formatiert werden sollen (denn dies kann ein Webbrowser nicht selbst entscheiden).

So kommt es, dass man mit Hilfe der HTML-Markups oder Tags, wie sie auch genannt werden, einen elektronisch erfassten Text schnell für die Anzeige in einem Webbrowser formatieren kann.

Um beispielsweise in unserem Kiefer-Beispiel dafür zu sorgen, dass die erste Zeile als Überschrift formatiert wird, braucht man den Text lediglich in die HTML-Tags `<h1>` und `</h1>` einzuschließen:

```
<h1>Die Kiefer in der chinesischen Malerei </h1>
```

Wie der Text dann aber letztendlich von den verschiedenen Browser dargestellt wird, in welcher Schriftart, welcher Größe und Farbe, darauf hat man selbst keinen Einfluss mehr. (Man kann aber davon ausgehen, dass der Text in großer, fetter Schrift und mit größerem Zeilenabstand angezeigt wird.)

Die Trennung von logischer Auszeichnung im HTML-Text und Formatierung durch den Webbrowser dient aber nicht nur der Bequemlichkeit, sie löst auch das Problem der Plattformunabhängigkeit (das Anfang der Neunziger von weit größerer Bedeutung war als heute). Als Webautor wissen Sie nichts über die Soft- und Hardware-Ausstattung der Internet-Teilnehmer, die Ihre Seiten ansteuern und lesen wollen. Arbeitet er mit einem textbasierten Betriebssystem oder verwendet er schon einen Windowing-Manager, verfügt er über einen 640x480-Bildschirm oder schafft seine Grafikkarte bereits Auflösungen von 1024x768 und höher, hat er einen Schwarzweiß- oder einen Farbmonitor, wie viele Farben schafft sein Display, welche Schriftarten sind auf seinem Rechner installiert, stehen TrueType-Schriften zur Verfügung? Explizite Formatierungsanweisungen, wie zum Beispiel die genaue Vorgabe einer zu verwendenden Schriftart, -farbe und -größe führen nur allzu schnell dazu, dass die Formatierung gänzlich verloren geht, wenn die gewünschten Formatierungen auf dem System des Webbrowsers nicht unterstützt werden (die Schriftart nicht vorhanden ist, Farben nicht angezeigt werden können). Logische Auszeichnungen wie `<h1>` können dagegen von allen Browsern verarbeitet und entsprechend ihrer Möglichkeiten umgesetzt werden. Die Überschrift ist dann vielleicht nicht ganz so perfekt, dafür aber auf jeden Fall als solche zu erkennen.

Es gibt noch einen Punkt, in dem die Erfinder von HTML den Browsern freie Hand ließen: der Festlegung der Zeilenumbrüche. Wenn Sie mit einem Textverarbeitungsprogramm wie Word einen Text zum Ausdrucken auf eine DIN-A4-Seite aufsetzen, richten Sie Word so ein, dass es alle Zeilen nach ungefähr 13 cm umbricht (so dass die Zeilen in etwa die Breite der DIN-A4-Seite (abzüglich Rand) ausfüllen, aber nicht über den Seitenrand hinausgehen). Sie können dies tun, weil Sie wissen, wie breit eine DIN-A4-Seite ist. Webseiten werden jedoch nicht ausgedruckt, sie werden in einem Webbrowser angezeigt. Die Breite des Browserfensters hängt aber davon ab, wie der Anwender das Browserfenster einstellt und welche Bildschirmauflösung er verwendet. Aus diesem Grunde überlässt es HTML den Browsern, Fließtexte entsprechend der aktuellen Breite des Browserfensters umzuberechnen.¹



Heute ist die plattformunabhängige Formulierung des HTML-Codes nicht mehr von der gleichen Bedeutung wie früher. Zum einen kann man davon ausgehen, dass heute fast alle Websurfer über Rechner mit Windowing-System und grafischem Browser, über Bildschirmauflösungen von 800x600 oder höher, 16-Bit-Farbaufösungen und einer größeren Auswahl an Standardschriften verfügen. Zum anderen haben Design und Erscheinungsbild der Webseiten gegenüber dem Inhalt immer mehr an Bedeutung gewonnen. Neu hinzugekommene HTML-Techniken, allen voran die Verwendung von Stylesheets (siehe Kapitel 3), lassen dem Webdesigner wesentlich mehr Gestaltungsfreiheit und räumen ihm größeren Einfluss über das endgültige Erscheinungsbild seiner Webseiten ein.

Formatierung mit HTML-Tags

In HTML werden Markups in Form von »Tags« direkt in den zu formatierenden Text eingefügt.

Die Tags

Jedes Tag besteht aus

- einer öffnenden Klammer `<`,
- einem Tag-Namen und
- einer schließenden rechten Klammer `>`.

Zum Beispiel:

```
<body>  
<h1>  
<p>  
...
```



HTML unterscheidet traditionell nicht zwischen Groß- und Kleinschreibung. Das Body-Tag, das den eigentlichen Seiteninhalt kennzeichnet, könnte man also sowohl als `<body>` wie auch als `<Body>` oder `<BODY>` schreiben. Der Browser wird es auf jeden Fall erkennen und richtig interpretieren. Andererseits erleben wir derzeit die Verschmelzung von HTML und XML. So existiert bereits seit Januar 2000 neben dem aktuellen HTML-4-Standard ein erster XHTML-Standard (XHTML steht für »eXtensible HTML«), in dem der aktuelle HTML-Standard in XML-konformer Weise redefiniert ist. Da XML zwischen Groß- und Kleinschreibung unterscheidet, sind gemäß XHTML 1.0 alle Tags (und Attribute) klein zu schreiben. Auch wenn Sie derzeit nicht daran interessiert sind, aus der Verschmelzung von XML und HTML Nutzen zu ziehen (siehe Kapitel 21), schadet es nichts, sich eine strikte Kleinschreibung anzugewöhnen.

HTML-Elemente

Aus Sicht von HTML besteht eine Webseite aus einer Abfolge von nacheinander aufgeführten, gegebenenfalls verschachtelten HTML-Elementen. Mit Ausnahme einiger weniger HTML-Elemente, die wir bei der Besprechung des HTML-Grundgerüsts kennen lernen werden, korrespondieren diese HTML-Elemente zu den verschiedenen Elementen der darzustellenden Seite (Absätze, Listen, spezielle formatierte Textpassagen, Bilder, etc.).

```
<h1>Die Kiefer in der chinesischen Malerei</h1>  
<hr />  
<p>In der traditionellen chinesischen Malerei gehört die Kiefer zu den am häufigsten dargestellten Bäumen überhaupt. Diese Vorrangstellung hat sie ihrer Genügsamkeit und ihren immergrünen Nadeln zu verdanken, die sie in der chinesischen Vorstellung zu einem Symbol für langes Leben und Beständigkeit werden ließen. So gehört die Kiefer auch zu den <em>drei Freunden im Winter</em>, den langes Leben symbolisierenden Pflanzen Kiefer, Bambus und Pflaume. ...</p>
```

Obiger Quelltextauszug enthält beispielsweise vier HTML-Elemente

- die Überschrift, die durch das `<h1>`-Tag gekennzeichnet ist.
- eine horizontale Trennlinie über die ganze Webseite, die durch das `<hr>`-Tag gekennzeichnet ist
- ein Textabsatz, der mit dem `<p>`-Absatz gekennzeichnet ist
- eine im Absatz eingeschlossene Textpassage, die durch das Tag `` gekennzeichnet ist.

Jedes Element beginnt mit einem Tag, das anzeigt, um was für ein Element es sich handelt, und dem Browser Hinweise auf dessen Formatierung gibt. Manche HTML-Elemente sind leer - wie zum Beispiel `<hr />`, das eine horizontale Linie repräsentiert. Die meisten HTML-Elemente haben jedoch einen Inhalt. In diesem Fall besteht das Element aus dem Start-Tag, dem nachfolgenden Inhalt und dem abschließenden Ende-Tag, das durch den Querstrich im Tag-Namen gekennzeichnet ist:

<h1>Die Kiefer in der chinesischen Malerei</h1>



Gemäß HTML-Spezifikation kann man leere Elemente wie <hr /> auch als <hr> oder <hr></hr> schreiben. Wir möchten Ihnen aber empfehlen, die Schreibweise <hr /> zu verwenden (Achtung: Nicht das Leerzeichen vor dem / vergessen!), da diese nicht nur zu HTML, sondern auch zu XHTML (siehe Kapitel 21) kompatibel ist. Aus den gleichen Gründen sollten Sie darauf achten, dass Sie alle nicht-leeren Elemente mit einem Ende-Tag abschließen.

Attribute

In den meisten Tags können zusätzliche Optionen als Attribute angegeben werden. Grundsätzlich sieht die Syntax für alle Tags in HTML wie folgt aus:

```
<tag attribut1="wert1" attribut2="wert2" ... > </tag>  
<tag attribut1="wert1" attribut2="wert2" ... />
```

Zum Beispiel:

```

```

Damit wird das Bild aus der Datei bild.tif in die Webseite eingefügt. In der Webseite wird für das Bild ein Bereich von 200 mal 500 Pixeln reserviert.



Und noch ein Hinweis zur XHTML-Kompatibilität: Die meisten Browser können Attribut-Werte auch ohne Anführungszeichen erkennen. Wer jedoch XHTML-kompatibel bleiben möchte, darf auf die Anführungszeichen nicht verzichten.

Zeilenumbrüche und Codierstil

Zeilenumbrüche, Tabulatoren und mehrere aufeinanderfolgende Leerzeichen, die Sie in den HTML-Code Ihrer Webseiten einfügen, sind für das Erscheinungsbild der Webseite im Browser nicht maßgeblich.

Für uns als Webautoren bedeutet dies, dass wir den HTML-Code der Webseiten durch Einfügen von Umbrüchen und mehrfachen Leerzeichen gestalten...

```
<html>  
<head>  
<title>Die Kiefer</title>  
</head>  
<body>  
<h1>Die Kiefer in der chinesischen Malerei </h1>  
<hr />
```

```
<p>In der traditionellen chinesischen Malerei gehört die Kiefer zu den am häufigsten dargestellten Bäumen überhaupt. Diese Vorrangstellung hat sie ihrer Genügsamkeit und ihren immergrünen Nadeln zu verdanken, die sie in der chinesischen Vorstellung zu einem Symbol für langes Leben und Beständigkeit werden ließen. So gehört die Kiefer auch zu den <em>drei Freunden im Winter</em> &#150; den langes Leben symbolisierenden Pflanzen Kiefer, Bambus und Pflaume. ...</p>
```

```
</body>
```

```
</html>
```

...oder durch Weglassen von Umbrüchen und mehrfachen Leerzeichen komprimieren können (um etwas Speicherplatz zu sparen, in diesem Beispiel in etwa 30 Byte, oder um Nachahmern das Leben zu erschweren):

```
<html><head><title>Die Kiefer</title></head><body><h1>Die Kiefer in der chinesischen Malerei </h1><hr /><p>In der traditionellen chinesischen Malerei gehört die Kiefer zu den am häufigsten dargestellten Bäumen überhaupt. Diese Vorrangstellung hat sie ihrer Genügsamkeit und ihren immergrünen Nadeln zu verdanken, die sie in der chinesischen Vorstellung zu einem Symbol für langes Leben und Beständigkeit werden ließen. So gehört die Kiefer auch zu den <em>drei Freunden im Winter</em> &#150; den langes Leben symbolisierenden Pflanzen Kiefer, Bambus und Pflaume. ...</p></body></html>
```

Kommentare

Mit Hilfe der Zeichenfolgen `<!--` und `-->` kann man Kommentare in den HTML-Code einfügen. Kommentare werden grundsätzlich vom Browser ignoriert, das heißt, der Text eines Kommentars wird nicht angezeigt.

```
<h1>Die Kiefer in der chinesischen Malerei</h1>
<!-- erstellt am 03.11.98 -->
<!-- Der nachfolgende Text basiert auf einem Artikel aus dem
      "Lexikon der chinesischen Symbole" von Wolfram Eberhard -->
<p>In der traditionellen chinesischen Malerei gehört die Kiefer zu den am häufigsten dargestellten Bäumen überhaupt. Diese Vorrangstellung hat sie ihrer Genügsamkeit ...
```

2.2 Das HTML-Grundgerüst

Sie haben nun schon eine ganze Menge über HTML erfahren. Warum legen wir nun nicht einfach los und schauen uns an, wie man einen Text in eine Webseite verwandelt - zumal es doch so einfach ist?

Nehmen wir den folgenden Text:

```
Die Kiefer in der chinesischen Malerei
In der traditionellen chinesischen Malerei gehört die Kiefer zu den am häufigsten dargestellten Bäumen überhaupt...
```

Um diesen Text in eine Webseite zu verwandeln, brauchen Sie ihn nur in einen geeigneten ASCII-Editor einzutippen und als Datei mit der Extension `.html` abzuspeichern. Beachten Sie, dass dieser Text keinen HTML-Code enthält. Trotzdem wird ihn der Browser mit hoher Wahrscheinlichkeit anzeigen. In einem weiteren Schritt könnten wir den Text mit HTML-Tags formatieren: die Überschrift in `<h3>`-Tags und den nachfolgenden Absatz in `<p>`-Tags einschließen.

<h3>Die Kiefer in der chinesischen Malerei</h3>

<p>In der traditionellen chinesischen Malerei gehört die Kiefer zu den am häufigsten dargestellten Bäumen überhaupt...</p>

Auch diese »Webseite« wird der Browser anzeigen, vermutlich sogar unter Berücksichtigung der von uns vorgegebenen Formatanweisungen. Dies liegt aber keineswegs daran, dass wir hier gültigen HTML-Code erzeugt hätten. Vielmehr ist es so, dass die meisten Webbrowser dermaßen fehlertolerant sind, dass sie selbst arg verstümmelten HTML-Code auszuwerten und den Dokumentinhalt anzuzeigen versuchen.

Natürlich darf man sich beim Aufsetzen der Webseiten nicht auf die Fehlertoleranz der Webbrowser verlassen. Vielmehr sollte man stets bemüht sein, korrekten HTML-Code aufzusetzen (möglichst nach der neuesten HTML-Spezifikation). Dazu gehört auch, dass jede Webseite über ein HTML-Grundgerüst verfügt. Büffeln wir also noch ein wenig Theorie und Formalismen, bevor wir im nächsten Abschnitt endgültig zum kreativen Teil übergehen.

Das Grundgerüst eines HTML-Dokuments besteht aus drei Teilen:

- einer Angabe zur HTML-Version

und - eingefasst in <html>-Tags -

- einem Head-Abschnitt mit Informationen und Deklarationen zum aktuellen Dokument
- einem Body-Abschnitt, der den eigentlichen anzuzeigenden Inhalt einschließt

Listing 2.1: Das HTML-Grundgerüst

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Das HTML-Grundgerüst</title>
  </head>
  <body>
    <p>Hallo vom Browser!</p>
  </body>
</html>
```



Die Aufteilung in Dokumentinformationen und anzuzeigenden Inhalt ist für elektronische Dokumente ganz typisch und weiter verbreitet als es den Anschein hat. Legen Sie doch einmal mit Ihrem bevorzugten Textverarbeitungssystem (Word oder StarWriter) ein neues, leeres Dokument an und speichern Sie es ab. Auf meinem System belegt ein leeres Word-Dokument immerhin 19 KByte. Nicht schlecht, wenn man bedenkt, dass die ersten PCs überhaupt nur 64 KByte Arbeitsspeicher hatten. Der Grund ist, dass Word in den doc-Dateien ebenfalls eine ganze Reihe von Metainformationen zu dem Dokument abspeichert (vornehmlich Informationen über die verwendeten Formatvorlagen). Wenn Sie das Dokument in Word laden, sehen Sie davon aber nichts, weil Word Ihnen nur den eigentlichen Inhalt anzeigt. Analog zeigen Webbrowser nur den Inhalt zwischen den <body>-Tags an.

Die Versionsinformation

An der Versionsinformation kann der Browser ablesen, gemäß welcher HTML- Spezifikation die vorliegende Webseite erstellt wurde. Der URL `http://www.w3.org/TR/html4/strict.dtd` weist zu einer Dokumententypdeklaration-Datei, hier `strict.dtd`, in der beschrieben ist, welche HTML-Elemente wie verwendet werden dürfen. Derzeit gibt es drei Dokumententypdeklaration-Dateien:

- *strict.dtd* - Die Standard-DTD, die für sauberen HTML 4.0-Code steht. Sie enthält keine veralteten Elemente und greift zur physikalischen Formatierung auf Stylesheets zurück.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
```

- *loose.dtd* - Eine Übergangs-DTD. Zusätzlich zu den Elementen aus *strict.dtd* enthält diese Dokumententypdeklaration eine Reihe von älteren Tags und Attributen, die durch die Einführung von Stylesheets oder neuerer Elemente an sich überflüssig geworden sind und eigentlich nicht mehr verwendet werden sollten (»deprecated« Elemente).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
```

- *frameset.dtd* - DTD für Webseiten, die in Frames unterteilt sind.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
    "http://www.w3.org/TR/html4/frameset.dtd">
```

Derzeit macht es kaum einen Unterschied, ob Sie Versionsinformationen angeben oder nicht. Die meisten Browser gehen, selbst wenn Sie die Versionsinformationen und die DTD-Datei auswerten, mit fehlerhaftem HTML-Code mehr als tolerant um - ein Verhalten, das noch aus der Zeit stammt, da es keine DTDs und damit keine festen Regeln für den Aufbau von HTML-Dokumenten gab. Im Hinblick auf die zukünftige Entwicklung sollte man jedoch stets Versionsinformationen und DTD-Datei angeben.



Soweit nicht explizit anders angegeben, entsprechen die HTML-Beispiele in diesem Buch alle der HTML 4.01/strict.dtd-Spezifikation.



Noch fortschrittlicher ist es, sich den XHTML-Regeln zu unterwerfen (siehe Kapitel 21).

Der Header-Abschnitt

Im Header können Sie allgemeine Informationen über die aktuelle Webseite unterbringen: Informationen über den Autor der Webseite, Copyright-Informationen, Informationen für Suchmaschinen und -roboter, etc. Die HTML-Spezifikation stellt uns dazu das `<meta>`-Tag zur Verfügung, mit dem wir uns in Kapitel 6

näher beschäftigen werden.

Welche <meta>-Elemente Sie in den Header einer Webseite aufnehmen, bleibt ganz Ihnen überlassen. Das zweite HTML-Element, das innerhalb des Headers verwendet werden kann, ist dagegen obligatorisch: es ist das <title>-Element, mit dem Sie einen Titel für Ihre Webseite angeben. Die meisten Browser zeigen den Titel in der Titelleiste Ihres Fensters an.

```
<head>
  <title>Dies ist der Titel der Webseite</title>
  <meta name="Author" content="Louis, Wenz">
</head>
```



Ebenfalls in den Header gehört das <base>-Tag, das wir Ihnen im Kapitel 6 näher vorstellen werden.

Der Body-Abschnitt

Die <body>-Tags umschließen den wichtigsten Teil Ihres HTML-Dokuments: den anzuzeigenden Inhalt. Alle nachfolgend beschriebenen HTML-Elemente, mit denen wir Text, Bilder, Hyperlinks u.a.m. in unsere Webseite aufnehmen und formatieren, stehen also innerhalb der <body>-Tags.

Damit wissen wir jetzt auch, wie aus dem HTML-Fragment vom Beginn dieses Abschnitts ein echtes HTML-Dokument wird:

Listing 2.2: kiefer.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Die Kiefer</title>
</head>
<body>

<h1>Die Kiefer in der chinesischen Malerei </h1>
<p>In der traditionellen chinesischen Malerei gehört die Kiefer zu den am
häufigsten dargestellten Bäumen überhaupt. Diese Vorrangstellung hat sie ihrer
Genügsamkeit und ihren immergrünen Nadeln zu verdanken, die sie in der
chinesischen Vorstellung zu einem Symbol für langes Leben und Beständigkeit
werden ließen. So gehört die Kiefer auch zu den <em>drei Freunden im
Winter</em> &#150; den langes Leben symbolisierenden Pflanzen Kiefer, Bambus
und Pflaume. ...</p>
</body>
</html>
```



Abbildung 2.3: Unsere erste Webseite

2.3 Text und Überschriften

Für die Aufbereitung des Textes im Body-Bereich stehen eine Vielzahl von Tags zur Verfügung, die in diesem und den nachfolgenden Abschnitten beschrieben werden.

Überschriften

Zur Hervorhebung von Überschriften stellt uns HTML sechs Tags zur Verfügung: <h1> bis <h6> (h steht hierbei für heading = Überschrift). Die sechs Überschriftenformate unterscheiden sich in Größe und Fettdruck der verwendeten Schrift.

| Tag | Darstellung im Internet Explorer |
|------|----------------------------------|
| <h1> | Dies ist Überschrift 1 |
| <h2> | Dies ist Überschrift 2 |
| <h3> | Dies ist Überschrift 3 |
| <h4> | Dies ist Überschrift 4 |
| <h5> | Dies ist Überschrift 5 |
| <h6> | Dies ist Überschrift 6 |

Tabelle 2.1: HTML-Überschriften



HTML-Überschriften sind grundsätzlich nicht nummeriert. Seit der CSS2-Spezifikation gibt es aber die Möglichkeit, die Überschriften bei Bedarf mit Hilfe von Stylesheets automatisch durchnummerieren zu lassen (allerdings wird diese Technik derzeit noch kaum von den Browsern unterstützt).

Inhaltsverzeichnisse

Klassische Inhaltsverzeichnisse sind für Webseiten eher ungewöhnlich. Trotzdem spricht nichts dagegen, größeren Texten ein Inhaltsverzeichnis voranzustellen. Da in einem solchen Inhaltsverzeichnis üblicherweise die ersten zwei oder drei Überschriftsebenen aufgelistet werden, stellt sich die Frage, ob man ein solches Inhaltsverzeichnis nicht aus den HTML-Überschriften automatisch generieren lassen kann? Nun, mit HTML allein geht dies nicht, vielleicht kann es aber Ihr Webeditor (FrontPage 2000 enthält beispielsweise eine Funktion, die für ein Web ein Inhaltsverzeichnis erstellt, in dem die einzelnen Seiten des Webs als Verzeichniseinträge aufgelistet werden). Falls nicht, müssen Sie notgedrungen selbst Hand anlegen. Was dabei grundsätzlich zu beachten ist, erfahren Sie in Abschnitt 2.6.4.

Absätze

Fließtexte werden traditionell in Absätze aufgeteilt. Absätze lockern das Seitenbild auf (nichts ist furchtbarer als Buchseiten, die aus einem einzigen Block von Wörtern und Buchstaben bestehen) und erleichtern das Lesen (insbesondere, wenn die Aufteilung in Absätze die Gedankensprünge im Inhalt widerspiegelt).

Absätze beginnen grundsätzlich am Anfang einer neuen Zeile, bestehen aus einer oder mehreren Zeilen Fließtext, die automatisch am Seitenrand umgebrochen werden, und enden mit einem expliziten Zeilenumbruch. Zudem ist der Abstand zwischen Absätzen meist etwas größer als der Zeilenabstand innerhalb der Absätze.

In den meisten Textverarbeitungssystemen drückt man zum Abschluss eines Absatzes die (Return)-Taste. Zur Kennzeichnung von Absätzen im HTML-Code reicht dies aber nicht aus, da die Webbrowser den resultierenden Zeilenumbruch nur als einfachen Leerraum, und nicht als Absatzende interpretieren. Aus diesem Grund müssen in HTML alle Absätze in die Tags `<p>` und `</p>` eingefasst werden (p steht für paragraph = Absatz).

```
<p>Dies ist ein Beispiel für einen Absatz. Eingegebene RETURNS werden ignoriert. Ein Umbruch erfolgt erst dann, wenn der Absatz mit dem Ende-Tag abgeschlossen wird.</p>
```



Setzen Sie keine Leerzeichen nach einem Start-Tag oder vor einem End-Tag, da diese von den Browsern unter Umständen ignoriert werden.

Zeilenumbrüche erzwingen

Wenn Sie einen Zeilenumbruch innerhalb eines Absatzes erzwingen wollen, setzen Sie das `
`-Tag. So wird der folgende HTML-Code

```
<p> Hier beginnt ein neuer Abschnitt. Ein Zeilenumbruch <br />kann auch mit dem BR-Tag erzwungen werden.</p>
```

im Browser wie folgt angezeigt:

Hier beginnt ein neuer Abschnitt. Ein Zeilenumbruch kann auch mit dem BR-Tag erzwungen werden.

Leerzeilen werden ebenfalls mit dem `
`-Tag erzeugt.



Schließlich gibt es noch die Möglichkeit, einen Fließtext mit Hilfe des `<pre>`-Tags als vorformatiert zu kennzeichnen. Dies weist den Webbrowser an, den eingeschlossenen Text so wiederzugeben, wie er im HTML-Code steht (unter Berücksichtigung aller Leerzeichen, Tabulatoren und (Return)-Zeilenumbrüche).

Text einrücken

Um einen oder mehrere ganze Absätze einrücken zu lassen, fasst man Sie in `<blockquote>`-Tags ein:

```
<h2>Sonnenkollektoren</h2>
<p>Sonnenkollektoren dienen dazu, die Energie aus der Sonneneinstrahlung in nutzbare Wärmeenergie umzuwandeln. Dass man auf diese Weise kostengünstig zu einem ausgeglichenen Wärmehaushalt kommen kann, beweist das Überleben einer Jahrtausende alter Spezies:
<blockquote><p>Die Haare des Eisbären erscheinen deswegen weiß, weil sie transparente, nicht pigmentierte Fasern darstellen, in denen UV-Licht wirkungsvoll zur Haut geleitet wird. Die Haut des Eisbären ist schwarz und somit bestens geeignet die Strahlungsenergie des UV-Lichtes aufzunehmen. Dass lediglich UV-Licht absorbiert wird, ist verständlich, denn würde der Eisbär versuchen, im sichtbaren Bereich zu absorbieren, müsste er schließlich seine weiße Tarnung opfern. UV-Licht hat aber immerhin den Vorteil, dass es auch bei bedecktem Himmel verfügbar ist und vom Eisbären mit einem Wirkungsgrad von 95% in Wärme umgesetzt werden kann.</p></blockquote>
<p>Sonnenkollektoren sind ganz ähnlich aufgebaut:</p>
```

Wie dieser Text in einem Webbrowser dargestellt wird, sehen Sie in Abbildung 2.4.

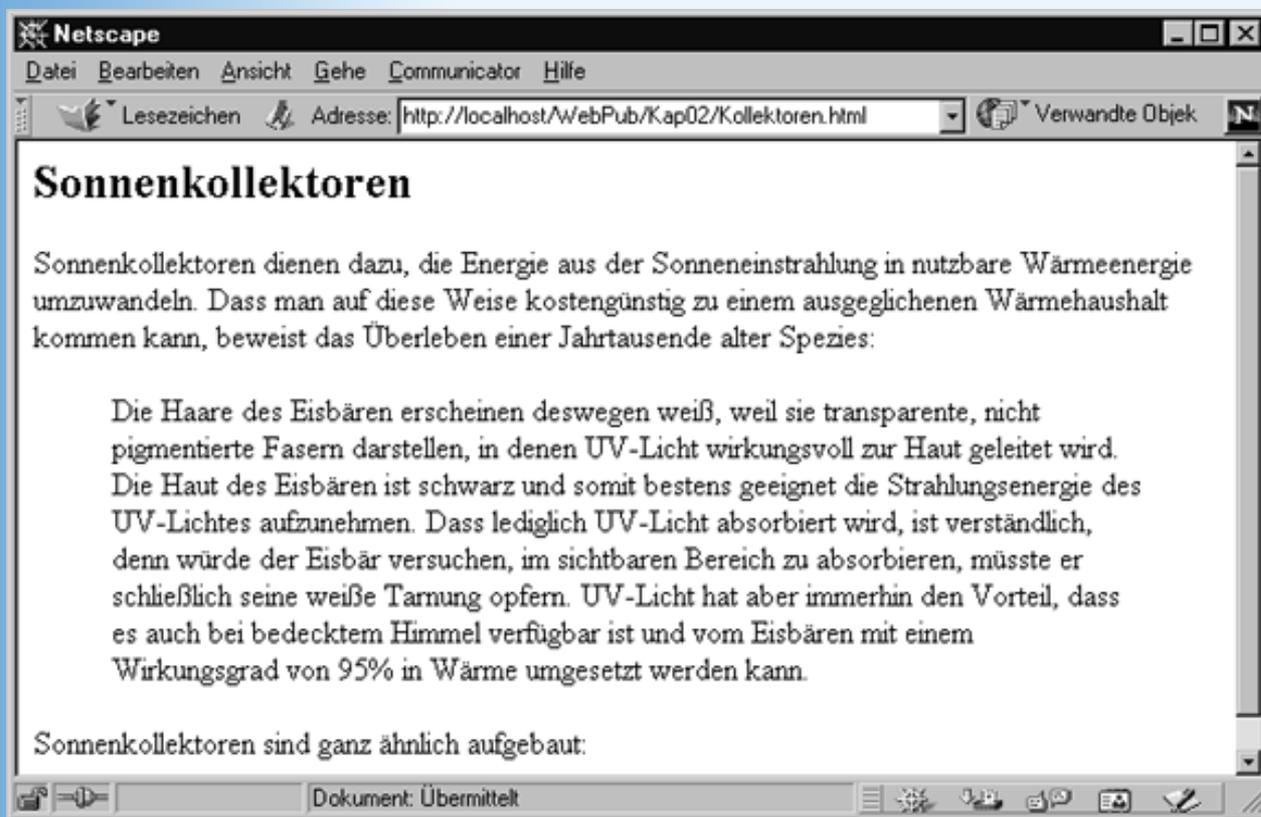


Abbildung 2.4: Einrücken mit `<blockquote>`

Das `<blockquote>`-Tag ist insbesondere zum Einrücken von größeren Zitaten gedacht. Stammt das Zitat von einem Webdokument, kann man die Adresse des Dokuments als Wert des `cite`-Attributs angeben (die aber nicht von allen Browsern angezeigt wird):

```
<blockquote cite="http://www.fiktiv.com/eisbaer.html"><p>Die Haare ...
```

Zum Einrücken beliebiger Textblöcken sollte man auf die `style`-Eigenschaften `padding` oder `margin` zurückgreifen (siehe Abschnitt 4.3.2). Dies hat unter anderem den Vorteil, dass man selbst bestimmen kann, wie weit der Absatz eingerückt werden soll.

Textstellen einzeln formatieren

Zur Auszeichnung von Textpassagen gibt es zwei Kategorien von Formatierungen:

- physikalische Formatierungen und
- logische Formatierungen.

Physikalische Formatierungen

Physikalische Formatierungen schreiben dem Webbrowser explizit vor, wie der Text aussehen soll.

| Tag | Beschreibung |
|---|----------------|
| <code> ... </code> | Fettschrift |
| <code><i> ... </i></code> | Kursivschrift |
| <code><small> ... </small></code> | kleine Schrift |
| <code><big> ... </big></code> | große Schrift |

| | |
|------------------|-------------------------|
| <tt> ... </tt> | Schreibmaschinenschrift |
| _{...} | tiefer gestellter Text |
| ^{...} | höher gestellter Text |

Tabelle 2.2: Physikalische Auszeichnungen

HTML definiert nur wenige Tags zur physikalischen Formatierung. Wer mehr Kontrolle über die Formatierung seiner Webseiten im Browser ausüben will, kann zur Formatierung mit Stylesheets greifen (siehe Abschnitt 2.8 und Kapitel 4).

Logische Formatierungen

Logische Schriftauszeichnungen werden dazu verwendet, inhaltlich verwandte Textstellen besonders zu kennzeichnen - beispielsweise Zitate, Definitionen, Listings. Mit der logischen Auszeichnung ist meist auch eine besondere optische Hervorhebung verbunden, die allerdings standardmäßig vom Browser ausgewählt wird.

| Tag | Beschreibung |
|------------------------|---|
| <abbr>...</abbr> | Abkürzung

Das title-Attribut ¹ des Elements kann zum Angeben des vollständigen Begriffes genutzt werden.

<p>Der <abbr title="World Wildlife Fund">WWF</abbr> legt großen ...</p> |
| <acronym>...</acronym> | Akronym

Das title-Attribut ² des Elements kann zum Angeben des vollständigen Begriffes genutzt werden. |
| <cite>...</cite> | Zitat oder Referenz |
| <code>...</code> | Listing |
| <dfn>...</dfn> | Definition |
| ... | Hervorhebung, üblicherweise durch Kursivschrift |
| <kbd>...</kbd> | Tastatureingabe |
| <samp>...</samp> | Beispiel |
| ... | Hervorhebung, üblicherweise durch Fettschrift |
| <var>...</var> | Variable (eines Programms, einer mathematischen Formel) |

Tabelle 2.3: Logische Auszeichnungen

1

Das title-Attribut ist für alle Tags definiert, die im Body-Abschnitt des HTML-Dokuments verwendet werden können. Die Browser sind gehalten den Wert des title-Attributs vorzulesen (als Akustikausgabe) oder als Kurzinformation anzuzeigen, wenn der Mauszeiger über dem Element verweilt.

2

Das title-Attribut ist für alle Tags definiert, die im Body-Abschnitt des HTML-Dokuments verwendet werden können. Die Browser sind gehalten den Wert des title-Attributs vorzulesen (als Akustikausgabe) oder als Kurzinformation anzuzeigen, wenn der Mauszeiger über dem Element verweilt.

Logische Auszeichnungen haben zwei entscheidende Vorteile:

- logische Auszeichnungen können mehrere physikalische Formatierungen umschließen.
- logische Auszeichnungen machen Dokumente für die Weiterverarbeitung mit Programmen zugänglich

Nehmen Sie an, Sie setzen einen Text auf, in dem es um den therapeutischen Nutzen verschiedener pharmakologischer Wirkstoffe geht. Deren Namen sollen jeweils durch Kursivschrift, eine etwas größere Schrift und eine andere Textfarbe hervorgehoben werden. (Zugegeben, das ist ein bisschen zuviel des Guten, aber es geht uns ja hier um den didaktischen Effekt.) Sie könnten dies erreichen, indem Sie alle Namen in folgende Tag- Sequenzen einschließen:

```
<font color="red" size="12"><i>Acetylsäure</i></font>
```

Alle Namen pharmakologischer Wirkstoffe auf diese Weise zu formatieren, ist recht aufwendig. Wesentlich einfacher ist es, sich eine logische Auszeichnung auszuwählen (sagen wir `<var>`) und diese zur Kennzeichnung zu verwenden. Jetzt muss man nur noch im Header der HTML-Datei festlegen, welche Formatierungen die Webbrowser für die Auszeichnung `<var>` verwenden sollen (wie dies geht, erfahren Sie in Kapitel 4). Der springende Punkt hierbei ist, dass die Formatierungsanweisungen nur noch einmal (bei der Festlegung der Formatierung für das Tag) und nicht mehr bei jedem Vorkommen des Tags angegeben werden müssen.



Das ``-Tag, das wir in diesem Beispiel verwenden, gehört an sich zu den traditionellen, physikalischen HTML-Formatierungen. Aus oben genannten Gründen ist es aber als »deprecated« eingestuft und sollte nicht mehr verwendet werden.

Logische Auszeichnungen haben noch einen weiteren Vorteil: mit ihrer Hilfe kann man Textelemente trotz gleicher Formatierung unterscheiden.

Physikalische Auszeichnungen neigen dazu, Unruhe in das Erscheinungsbild eines Textes zu bringen. Nehmen wir an, Sie wollen im oben angesprochenen Text nicht nur die Namen der pharmakologischer Wirkstoffe, sondern auch die Namen der angesprochenen pharmazeutischen Firmen hervorheben. Um das Schriftbild ruhig zu halten, entscheiden Sie sich in beiden Fällen für eine dezente Hervorhebung: die

Kursivschrift. Wenn Sie jetzt sowohl die Namen der Wirkstoffe wie auch der Firmen in `<i>`-Tags erfassen, haben Sie unter Umständen ein Problem. Stellen Sie sich vor, Sie wollen ein Programm schreiben, das alle Namen pharmakologischer Wirkstoffe in Ihrem HTML-Text findet. Woran kann ein Programm erkennen, dass es sich bei einem Wort im Text um den Namen eines Wirkstoffes handelt? Nun, es könnte die Namen ganz einfach an den `<i>`-Tags erkennen, wenn Sie die `<i>`-Tags nicht auch zur Kennzeichnung der Firmennamen verwendet hätten. Damit liegt die Lösung auf der Hand: Sie suchen sich zwei logische Auszeichnungen, die üblicherweise von den Browsern kursiv formatiert werden - beispielsweise `` und `<var>` - und verwenden `` für die Firmennamen und `<var>` ausschließlich zur Kennzeichnung der Wirkstoffe. Schon kann das Programm die Wirkstoffe mühelos anhand der `<var>`-Tags erkennen.



Der Analyse von Dokumenten durch Programme kommt heute eine immer größere Bedeutung zu. Es wäre jedoch vermessen zu behaupten, dass diese Entwicklung auf der konsequenten Verwendung der logischen HTML-Auszeichnungen beruhen würde. Nein, getragen wird diese Entwicklung vor allem von XML. XML ist eine Markup-Language und so etwas wie der große Bruder von HTML. Was XML auszeichnet, ist, dass man in XML eigene Tags definieren und Regeln für den Aufbau der Dokumente festlegen kann. Mehr zu XML erfahren Sie in Kapitel 21.

Sonderzeichen

Wenn Sie im HTML-Text Sonderzeichen verwenden, beispielsweise die deutschen Umlaute, kann es sein, dass diese nicht in allen Browsern korrekt dargestellt werden. Aus diesem Grunde ist es vorzuziehen, diese Umlaute durch die entsprechenden HTML- Codierungen zu ersetzen.

| Sonderzeichen | Umschreibung |
|---------------|---|
| Leerzeichen | ¹ |
| < > & » | < > & " |
| ä ü ö ß | ä ü ö ß |
| Ä Ü Ö | Ä Ü Ö |
| à â è é ê ë ô | à â è é ë ô |
| α β γ δ usw. | α β γ δ usw. |

Tabelle 2.4: Codierung von Sonderzeichen

1

Folgen mehrere Leerzeichen im HTML-Code aufeinander, streicht sie der Browser bis auf ein Leerzeichen zusammen (Leerzeichen, die allein in einem Tag stehen, werden ganz entfernt). Will man dies verhindern, muss man die Leerzeichen als ` ` codieren.



Eine umfangreichere Liste finden Sie in Anhang B. Die komplette Liste entnehmen Sie bitte der HTML-Spezifikation (<http://www.w3.org>).



Der besseren Lesbarkeit der Listings zuliebe werden wir in den hier abgedruckten Beispielen aber vorwiegend die normalen deutschen Umlaute verwenden.

2.4 Listen

HTML definiert drei unterschiedliche Arten von Listen:

- Aufzählungen (ungeordnete Listen)
- Nummerierungen (geordnete Listen)
- Definitionen.

Aufzählungen

Aufzählungen sind Listen, in denen die einzelnen Listeneinträge durch vorangestellte Aufzählungszeichen (einem Punkt oder einen Quadrat) gekennzeichnet werden.

Aufzählungen werden verwendet, wenn es darum geht, ungeordnete Informationen, bei denen die Reihenfolge der Präsentation keine oder nur eine untergeordnete Rolle spielt, in übersichtlicher Form anzuzeigen. Einzelne Elemente der Liste können dabei ebenso gut aus einem einzigen Stichwort wie aus mehrzeiligen Absätzen (ja sogar aus mehreren Absätzen) bestehen.

...

Die gesamte Liste wird zunächst mit dem -Tag umschlossen (ul = unordered list = ungeordnete Liste).

...

Jedes einzelne Listenelement muss zusätzlich noch einmal in -Tags eingeschlossen werden (li = list item = Listenelement).

Beispiel:

```
<ul>
  <li>Aufzählungen (ungeordnete Listen)</li>
  <li>Nummerierungen (geordnete Listen)</li>
  <li>Definitionen</li>
</ul>
```

Nummerierungen

Nummerierungen werden verwendet, wenn es darum geht, geordnete Informationen zu präsentieren, die in einer bestimmten Abfolge, Rangfolge oder Wertung stehen. Einzelne Elemente der Liste können wie bei den Aufzählungen aus einem einzigen Stichwort oder aus mehrzeiligen Absätzen bestehen. Der Browser nummeriert die Listeneinträge automatisch.

...

Die gesamte Liste wird zunächst mit dem -Tag umschlossen (ol = ordered list = geordnete Liste).

...

Jedes einzelne Listenelement muss zusätzlich noch einmal in -Tags eingeschlossen werden (li = list item = Listenelement).

Beispiel:

```
<h2>Deutsche Bühnen</h2>
<p>Welche Bühnenautoren werden im deutschsprachigen Raum am häufigsten
gespielt?
<ol>
  <li>William Shakespeare</li>
  <li>Johann Wolfgang von Goethe</li>
  <li>Berthold Brecht</li>
  <li>Molière</li>
  <li>Friedrich Schiller</li>
  <li>Henrik Ibsen</li>
</ol>
```



Abbildung 2.5: Nummerierung im Webbrowser

Definitionen

Definitionen bestehen aus zwei Teilen:

- dem zu definierenden Begriff, dem die Formatvorlage Definierter Begriff zugewiesen wird und

- der nachfolgenden Definition des Begriffes, dem die Formatvorlage Definition zugewiesen wird.

Definitionen verwenden keine Listensymbole. Stattdessen steht der zu definierende Begriff linksbündig im Text und die nachfolgende Definition wird eingerückt.

<dl>...</dl>

Die gesamte Definitionsliste wird zunächst mit dem <dl>-Tag umschlossen (dl = definition list).

<dt>...</dt>

Die zu definierenden Begriffe werden in <dt>-Tags eingeschlossen (dt = definition term = Definitionsbegriff).

<dd>...</dd>

Die eingerückten Definitionen werden in <dd>-Tags eingeschlossen (dd = definition description = Definitionsbeschreibung).

Beispiel:

```
<h2>Unser Ensemble stellt sich vor:</h2>
<dl>
  <dt>Groucho Marx</dt>
  <dd>Unser Dirigent und Manager. </dd>
  <dt>&nbsp;</dt>
  <dt>Chico Marx</dt>
  <dd>Der Pianist.</dd>
  <dt>&nbsp;</dt>
  <dt>Harpo Marx</dt>
  <dd>Souverän an Harfe und Tröte.</dd>
  <dt>&nbsp;</dt>
  <dt>Zeppo Marx</dt>
  <dd>Unser Heldentenor.</dd>
  <dt>&nbsp;</dt>
  <dd>&nbsp;</dd>
</dl>
```

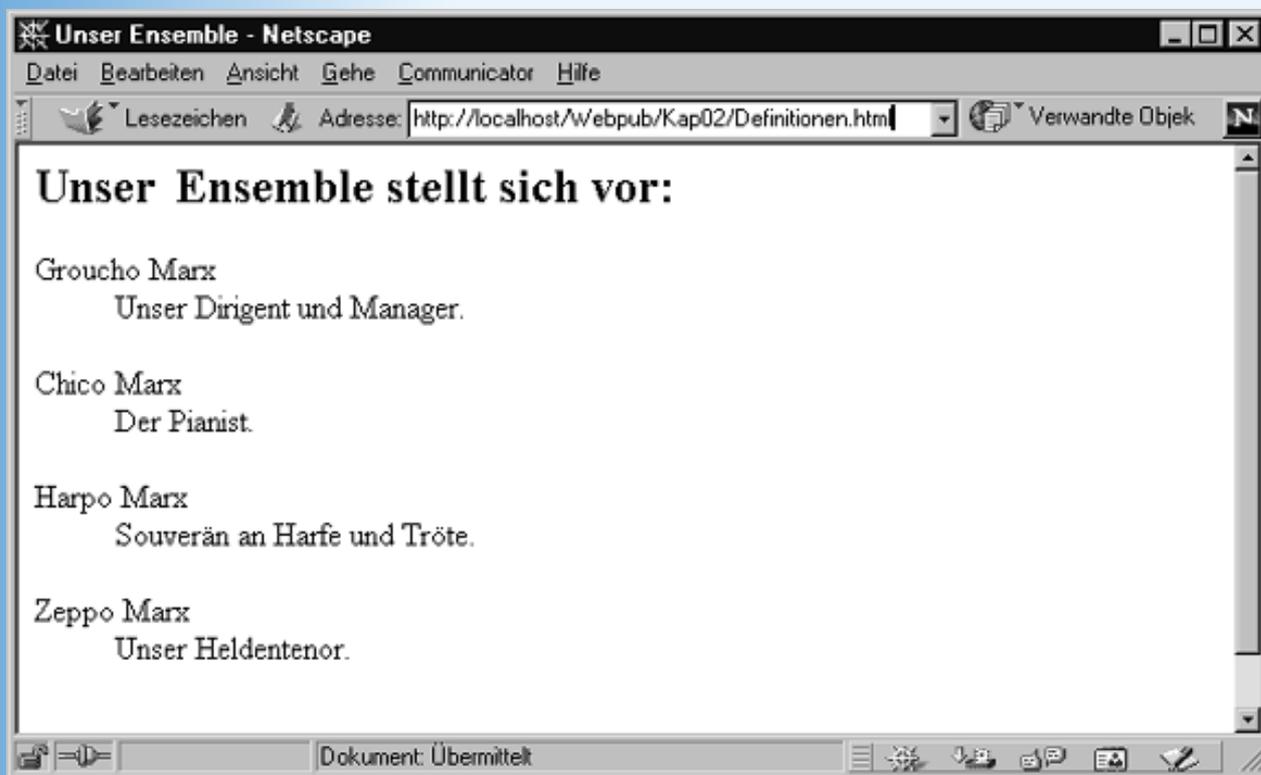


Abbildung 2.6: Definitionsliste im Webbrowser

Listen formatieren

Listen können in vielfältiger Weise formatiert und angepasst werden. Die meisten dieser Formatierungen beruhen auf Stylesheets. In Kapitel 4 werden wir uns mit diesen Formatierungsmöglichkeiten noch eingehend beschäftigen. Einige recht einfache Formatierungen möchte ich Ihnen aber jetzt schon vorstellen.

Aufzählungssymbole

Über die Stylesheet-Eigenschaft `list-style-type` können Sie zwischen drei Aufzählungssymbolen wählen:

- disc (gefüllter Kreis²)
- circle (leerer Kreis)
- square (gefülltes Quadrat)

Sie brauchen das Listen-Tag nur um ein `style`-Attribut zu erweitern und der `list-style-type`-Eigenschaft den gewünschten Wert zuweisen:

```
<ul style="list-style-type:square">  
  <li>Element1</li>  
  <li>Element2</li>  
  <li>Element3</li>  
</ul>
```

Nummerierungszeichen

Für geordnete Listen können Sie zwischen verschiedenen Nummerierungszeichen wählen:

- none (keine Nummerierung)
- decimal (1, 2, 3 ...)

- lower-alpha (a, b, c ...)
- upper-alpha (A, B, C ...)
- lower-roman (i, ii, iii, iv ...)
- upper-roman (I, II, III, IV ...)

Nach CSS2 sind sogar noch weitere Nummerierungen definiert, beispielsweise hebrew (hebräisch) oder katakana (japanisch).

Verschachtelte Listen

Listen können auch ineinander verschachtelt werden.

```
<ul>
  <li>Element1
    <ul>
      <li>Element11</li>
      <li>Element12</li>
    </ul>
  </li>
  <li>Element2
    <ul>
      <li>Element21</li>
      <li>Element22</li>
      <li>Element23</li>
    </ul>
  </li>
</ul>
```

Der Webbrowser sorgt dabei für die Einrückung der inneren Listen. Beachten Sie, dass der Webbrowser keine hierarchische Nummerierung für verschachtelte geordnete Listen erzeugen kann.



Die CSS2-Spezifikation für Stylesheets (siehe Kapitel 4) sieht zwar Techniken vor, wie man hierarchisch durchnummerierte Listen (1, 1.1, 1.1.1, etc.) erzeugen kann, doch werden diese derzeit von keinem Browser unterstützt.

Wenn Sie möchten, dass die untergeordneten Listenebenen vom Leser ein- und ausgeblendet werden können, müssen Sie ein passendes JavaScript-Skript aufsetzen (siehe Kapitel 10.7).

2.5 Bilder

Bilder sind heute aus dem Web gar nicht mehr weg zu denken. Dabei machen die Bilder, die in Form einer Galerie oder eines Internet-Fotoalbums präsentiert werden, noch den geringsten Teil aus. Dafür werden Bilder mehr und mehr beim Webdesign eingesetzt: als Schaltflächen-Links, Aufzählungssymbole, Hintergründe, Rahmen oder auch zur Gestaltung des Webseitenvordergrundes.

Das -Tag

Der übliche Weg zur Einbindung einer Grafik führt über das ``-Tag (`img` = `image` = `Bild`).

```

```

Das Attribut `src` gibt den URL der Bilddatei an. Üblicherweise gibt man hier einen relativen Pfad an, das heißt einen Pfad, der relativ zum Ursprungsort des Webdokuments angegeben wird (oder relativ zur Basisadresse des Dokuments, falls eine solche im Header- Abschnitt mit Hilfe des `<base>`-Tags spezifiziert wurde). Wenn obige HTML-Zeile beispielsweise in einem HTML-Dokument steht, das keine spezielle Basisadresse spezifiziert, geht der Browser davon aus, dass die Bilddatei *kiefer.tif* in dem gleichen Verzeichnis steht wie das HTML-Dokument. Wäre die Bilddatei in einem Unterverzeichnis */images* zu finden, würde man als relativen Pfad "images/kiefer.tif" angeben.

Zusätzlich zur Bildquelle sollte man stets einen kurzen, beschreibenden Text angeben (`alt`-Attribut). Dieser wird von reinen Textbrowsern als Bildersatz angezeigt. Grafische Browser blenden diesen Text ein, wenn der Anwender die Darstellung von Bildern deaktiviert hat oder die Bilddatei unter der angegebenen URL nicht gefunden wurde.



In früheren Versionen war das alt-Attribut nur optional zu benutzen, in HTML 4.0 ist seine Nutzung zwingend vorgeschrieben.

Bildabmaße

Mit den Attributen `width` und `height` kann man festlegen, welchen Raum das Bild auf der Webseite einnehmen soll (Angaben erfolgen standardmäßig in Pixel). Wenn die Abmaße der Bilddatei nicht mit den Angaben zu `width` und `height` übereinstimmen, skaliert der Browser das Bild entsprechend.



Beachten Sie, dass eine Skalierung meist zu einer Verschlechterung der Bildqualität führt.

Aber auch wenn Sie das Bild nicht skalieren, empfiehlt es sich, die Bildabmaße anzugeben. Der Browser kann dann beim Seitenaufbau den Platz für das noch zu ladende Bild freihalten (was den Aufbau beschleunigt).

Listing 2.3: kiefer2.html - Einbindung eines Bildes

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Die Kiefer</title>
</head>

<h1>Die Kiefer in der chinesischen Malerei</h1>
```

```
<p>In der traditionellen chinesischen Malerei gehört die Kiefer zu den am häufigsten dargestellten Bäumen überhaupt. Diese Vorrangstellung hat sie ihrer Genügsamkeit und ihren immergrünen Nadeln zu verdanken, die sie in der chinesischen Vorstellung zu einem Symbol für langes Leben und Beständigkeit werden ließen. So gehört die Kiefer auch zu den <em>drei Freunden im Winter</em> &#150; den langes Leben symbolisierenden Pflanzen Kiefer, Bambus und Pflaume. ...</p>
```

```
</body>
```

```
</html>
```



Abbildung 2.7: Webseite mit eingefügtem Bild

Positionierung und Textfluss

Traditionell gibt es in HTML eine Reihe von Attributen, mit denen man steuern kann, wie ein Bild in eine Seite eingefügt werden soll: `borders`, `hspace`, `vspace`, `align`. Alle diese Attribute sind jedoch mittlerweile »deprecated« und sollten nicht mehr verwendet werden. Macht nichts, es gibt ehedem bessere Möglichkeiten.

Tabellen sind beispielsweise ein bewährtes Mittel, um Bild- und Textelemente mehr oder weniger frei auf einer Webseite zu verteilen.

Noch mehr Möglichkeiten bietet die Formatierung und Positionierung mit Stylesheets. Wie Sie es schon gewohnt sind, werden wir Ihnen an dieser Stelle ein Beispiel zeigen, wie man auf einfache Weise mit Inline-Styles formatieren kann, und Sie ansonsten auf Kapitel 4 verweisen.

Um die Webseite aus Abbildung 2.7 etwas interessanter zu gestalten, wollen wir den etwas langweiligen Aufbau aus vorangehendem Bild und nachfolgendem Text aufbrechen. Das schmale, hochgestreckte Bild der Kiefer würde sich doch wunderbar als Randbild eignen. Der Text könnte von oben und links

eingerrückt werden, so dass er neben dem Stamm der Kiefer erscheint (siehe Abbildung 2.8).



Abbildung 2.8: Freies Design mit style-Attributen

1. Als Erstes bearbeiten wir die Abbildung der Kiefer. Das »U« aus Gras, Stamm und überhängendem Ast eignet sich wunderbar als Rahmen für den Text. Es ist allerdings noch etwas zu eng. Wir haben die Bilddatei daher in ein geeignetes Grafikprogramm geladen (im vorliegenden Fall reicht schon Paint Shop Pro, das man als Demoversion von <http://www.jasc.com> herunterladen kann) und die Spitze des überhängenden Astes ein wenig »zurückgeschnitten«. Jetzt muss noch der Stamm verlängert werden. Dazu haben wir die Bildabmaße nach unten verlängert und die untere Bildhälfte ab der Mitte des Stammes nach unten verschoben. Die entstandene Lücke haben wir mit einem Bildstreifen gefüllt, den ich aus der Höhe des Stammes kopiert habe.
2. Statt das Bild mit dem -Tag einzufügen, definieren wir es als Hintergrundbild:

```
<body style="background-image: url('kiefer_b.tif')"
```

Wir hätten auch das -Tag verwenden können, aber dann kann man die Stileigenschaften nicht so übersichtlich zusammen in einem Tag spezifizieren.

3. Hintergrundbilder werden von den Browser standardmäßig wie Kacheln ausgelegt, damit sie den ganzen Webseitenhintergrund füllen. Um dieses Verhalten zu unterbinden, setzen wir die Stileigenschaft background-repeat auf no-repeat:

```
<body style="background-image: url('kiefer_b.tif');  
            background-repeat: no-repeat; background-color: white;"
```

4. Schließlich sorgen wir dafür, dass der eigentliche Seiteninhalt (unser Text) vom linken und oberen Seitenrand so weit abgerückt werden, dass er im »U« der Kiefer angezeigt wird.

```
<body style="background-image: url('kiefer.tif');
background-repeat: no-repeat; background-color: white;
margin-left: 200; margin-top: 210">
```



Grundsätzlich sollten Sie, wenn Sie ein Hintergrundbild verwenden, dass den Browserhintergrund nicht ganz ausfüllt, auch die Hintergrundfarbe definieren (so dass sie der Hintergrundfarbe des Hintergrundbildes entspricht). Gehen Sie keinesfalls davon aus, dass der Hintergrund automatisch weiß ist.

Den vollständigen HTML-Code sehen Sie in Listing 2.4. Beachten Sie, dass dieses Design eine gewisse Mindestbreite des Browserfensters voraussetzt. Ist das Fenster zu schmal, wird der Text zu oft umgebrochen und überlappt mit den Blumen und dem Gras. Andererseits ist die Darstellung schon bei einer Fensterbreite von weniger als 800 Pixeln perfekt, und da man davon ausgehen kann, dass die meisten Anwender Bildschirmauflösungen von 800x600 und mehr verwenden, dürfen wir ruhigen Gewissens erwarten, dass der Anwender sein Browserfenster entsprechend einstellt.

Listing 2.4: kiefer3.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Die Kiefer</title>
</head>
<body style="background-image: url('kiefer.tif');
background-repeat: no-repeat; background-color: white;
margin-left: 200; margin-top: 210">
<h1>Die Kiefer in der chinesischen Malerei</h1>
<p>In der traditionellen chinesischen Malerei gehört die Kiefer zu den am
häufigsten dargestellten Bäumen überhaupt. Diese Vorrangstellung hat sie ihrer
Genügsamkeit und ihren immergrünen Nadeln zu verdanken, die sie in der
chinesischen Vorstellung zu einem Symbol für langes Leben und Beständigkeit
werden ließen. So gehört die Kiefer auch zu den <em>drei Freunden im
Winter</em> &#150; den langes Leben symbolisierenden Pflanzen Kiefer, Bambus
und Pflaume. ...</p>
</body>
</html>
```

Bilddateien für die Verwendung in Webseiten aufbereiten

Bilder für Webseiten zusammenstellen und aufzubereiten, ist eine Kunst für sich. Eine Kunst, die nebenbei auch eine sehr technische Seite besitzt, der wir uns nun zuwenden wollen.

Die wichtigsten Bildformate

Grafikformate gibt es wie Sand am Meer, doch nicht alle sind für Webseiten geeignet. Welche besonderen Eigenschaften sollte ein Grafikformat für Webbilder haben? Die Antwort liegt auf der Hand: Um die Kapazitäten des Netzwerks und die Nerven der Websurfer zu schonen, sollte ein Grafikformat bei

annehmbarer Bildqualität möglichst kleine Dateigrößen liefern. Gleichzeitig sollte das Grafikformat einfach auszuwerten und darzustellen sein, damit es von möglichst allen Browsern unterstützt wird. Unter diesen Gesichtspunkten haben sich drei Grafikformate als Standardformate für Webbilder etabliert.

GIF steht für »Graphic Interchange Format« und ist eine Entwicklung von CompuServe. GIF-Bilder werden nach dem LZW-Algorithmus komprimiert, für den die Firma Unisys die Patentrechte innehat. Mit diesem Algorithmus sind hohe Komprimierungsraten ohne Qualitätsverluste erreichbar.

Das GIF-Format erlaubt traditionell nur 256 Farben (es gibt Erweiterungen des Formats, die mehr Farben gestatten, doch werden diese nur von wenigen Programmen unterstützt). Für photorealistische Darstellungen scheidet das GIF-Format daher aus. Es ist aber bestens für Design-Elemente (Schaltflächen, Hintergründe, etc.) und Bilder geringer Farbtiefe geeignet.

GIF-Bilder können als »interlaced« oder als »non-interlaced« abgespeichert werden. »Non-Interlaced«-Bilder werden Zeile für Zeile an den Webbrowser geschickt und können von diesem erst dann angezeigt werden, wenn sie ganz geladen sind. »Interlaced«-Bilder werden paketweise übertragen, wobei die Pakete jeweils nur jede achte Zeile enthalten. Auf diese Weise erhält der Browser sofort ein Rohbild, das er anzeigen kann und das Paket für Paket verbessert wird.

Ein weiterer Vorteile des GIF-Format ist, dass man eine transparente Farbe definieren kann.

Wenn Sie Grafiken im GIF-Format speichern und Ihr Grafikprogramm sowohl das 87a- als auch das 89a-Format unterstützt, wählen Sie das neuere 89a-Format.

JPG oder JPEG steht für Joint Photographic Expert Group. Es zeichnet sich durch einen leistungsstarken, konfigurierbaren Komprimierungsalgorithmus aus, der allerdings nicht verlustfrei arbeitet. Je stärker die Komprimierung, um so größer die Qualitätsverluste.

JPG unterstützt 16 Millionen Farben (TrueColor) und mittlerweile auch den schrittweisen Bildaufbau im Browser (progressive JPG). Es eignet sich vor allem für größere Bilder und für Fotos.

PNG 1995 kündigten CompuServe und Unisys an, dass sie auf den GIF-LZW-Komprimierungsalgorithmus Lizenzgebühren erheben wollen. Als Antwort wurde das nicht-kommerzielle PNG-Format entwickelt, das in gewisser Weise, das Beste aus GIF und JPG in sich vereint. Das PNG-Format eignet sich für alle Arten von Bildern. Im Vergleich zu JPG erlaubt es allerdings keine fein abgestufte Bildkomprimierung (es verwendet wie GIF ein verlustfreies Komprimierungsverfahren).

Noch hat sich PNG nicht wirklich durchgesetzt. Zum einem liegt dies vermutlich daran, dass die Endbenutzer (Webautoren) nicht von der Lizenz betroffen sind, zum anderen daran, dass ältere Browser PNG nicht unterstützen und GIF und JPG einfach gute Alternativen sind.

Zur besseren Übersicht hier noch einmal die wichtigsten Unterscheidungskriterien.

| Kriterium | GIF | JPG | PNG |
|-----------|---------|----------------|---|
| Farbtiefe | 2 - 256 | 256 - 16,7 mio | 2 - 65536 (S/W)
2 - nahezu unendlich |

| | | | |
|--------------------------|---------------------------|---|---|
| Komprimierung | verlustfrei | abgestuft, mit zunehmenden Qualitätsverlusten | verlustfrei |
| Transparenz | ja | nein | ja |
| Schrittweise Darstellung | ja | ja | ja |
| | interlaced-Modus | progressive JPG | interlaced-Modus |
| Animationen | ja | nein | nein |
| Alpha-Kanal | nein | nein | ja |
| Verwendung | Design-Elemente | Bilder hoher Farbtiefe | Alle Bilder, aber bevorzugt Bilder geringer Farbtiefe und Größe |
| | Bilder geringer Farbtiefe | Fotorealistische Darstellungen | |
| | | Große Bilder (> 10 KByte) | |

Tabelle 2.5: Grafikformate für Webbilder



Für die Bearbeitung von Bildern gelten andere Regeln als für die Veröffentlichung. Entsprechend werden auch andere Ansprüche an das Grafikformat gestellt. Wichtiger als die Größe der Bilddatei und die Unterstützung von Transparent und Interlacing ist die verlustfreie Speicherung und Bearbeitung. So scheiden JPG und GIF meist von vorneherein aus: JPG erlaubt kein verlustfreies Abspeichern und GIF ist für Bilder mit mehr als 256 Farben ungeeignet. Brauchbar sind dagegen TIFF und PNG.

Bilddateigröße versus Bildqualität

Als verantwortlicher Webautor sollten Sie Bilddateien nie ins Web stellen, ohne zuvor zu prüfen, ob man nicht die Größe der Bilddateien reduzieren könnte. Dies gilt für die Veröffentlichung von Fotos ebenso wie für Design-Elemente oder Bilder zur Unterstützung dynamischer Webinhalte.

Nehmen wir an, Sie haben eine Fotovorlage, die Sie zur Verwendung auf Ihren Webseiten einscannen und aufbereiten wollen.

1. Scannen Sie das Bild ein.

Die interessanteste Einstellung ist dabei die Auflösung in dpi (Dots per Inch = Punkte pro Zoll). Je höher die Auflösung, um so mehr Bildpunkte werden pro Zoll eingescannt und um so höher ist die Bildqualität. Für die Anzeige auf dem Computerbildschirm ist die Auflösung aber mehr oder weniger unerheblich, denn der Monitor kann maximal 75 Dots (Pixel) pro Zoll anzeigen. (So schaffen 17 Zoll- Bildschirme nur maximale Auflösungen von 1280 x 1024; 1280 Pixel : 75dpi = 17 Zoll.) Wenn Sie mit einer höheren Auflösung einscannen, wird das Bild auf dem Monitor nicht schärfer, sondern größer. So gesehen sind 75 dpi absolut ausreichend.

Wenn Sie allerdings beabsichtigen, das Bild (oder Teile des Bildes) zu vergrößern, reduziert sich sofort die Bildschärfe, sofern Sie die Vergrößerung nicht bei der Auswahl

der dpi-Auflösung eingerechnet haben.

| Mögliche Vergrößerung | Auflösung beim Scannen |
|----------------------------|------------------------|
| 100 % (keine Vergrößerung) | 75 dpi |
| 200 % | 150 dpi |
| 400 % | 300 dpi |
| 800 % | 600 dpi |

Tabelle 2.6: Verhältnis dpi zu Vergrößerung

Wenn Sie sich unsicher sind, wählen Sie beim Scannen eine höhere dpi-Auflösung. Heruntersetzen können Sie die Auflösung später immer noch (geschieht in den meisten besseren Grafikprogrammen durch Skalierung oder Veränderung der Bildabmaße).

2. Speichern Sie das Bild in dem von Ihnen gewählten Bearbeitungsformat (beispielsweise TIFF).

Achten Sie darauf, dass Sie ein Grafikformat wählen, das verlustfrei abgespeichert wird und das auch von dem Grafikprogramm, in dem Sie das Bild weiterbearbeiten wollen, unterstützt wird.

3. Legen Sie eine Kopie an und bearbeiten Sie diese in Ihrem Grafikprogramm.

Jetzt haben Sie Gelegenheit, Schriftzüge einzufügen, zu retuschieren, Farben zu verfremden, Effekte auszuprobieren.

Führen Sie unbedingt alle Bearbeitungsschritte an der Kopie aus, damit Sie notfalls immer zum Original zurückkehren können. (Verlassen Sie sich nicht allzu sehr auf die Rückgängig-Funktion Ihres Grafikprogramms.)

4. Speichern Sie das fertig bearbeitete Bild und legen Sie auch von diesem eine Kopie an.
5. Reduzieren Sie zum Schluss die Größe der kopierten Bilddatei.

Auf die Dateigröße können wir auf verschiedenen Wegen Einfluss nehmen:

- über die Farbtiefe,
- über die Bildabmaße und
- über das verwendete Grafikformat/Komprimierungsverfahren.

Feste Spielregeln gibt es für die Reduzierung kaum, das heißt, man muss ausprobieren, wie man zu dem bestmöglichen Ergebnis kommt. Zwei Fallbeispiele - für ein Foto und eine Grafik - sollen die grundsätzliche Vorgehensweise verdeutlichen.



Abbildung 2.9: Ausgangsbild und unterschiedlich stark komprimierte JPG-Dateien

Betrachten wir zuerst das Bild *Lilly400.tif* aus Abbildung 2.9. Es wurde mit einer Auflösung von 400 dpi und einer Farbtiefe von 16,7 Millionen Farben (24 Bit pro Farbe) eingescannt. Eine weitere Bearbeitung ist nicht vorgesehen, also können wir gleich mit der Reduzierung der Bilddateigröße beginnen - was auch dringend notwendig ist, weil das Original fast 16 MByte einnimmt (1889×2856 (Pixel) * 24 (Bit pro Pixel)).

Zuerst einmal ist das Bild so viel zu groß (in Abbildung 2.9 ist das Bild um den Faktor 6 verkleinert). Wir skalieren es unter Beibehaltung des Seitenverhältnisses auf 315×476 herunter und schon ist die Datei nur noch 440 KByte groß.

Da es sich um ein relativ großes Foto mit vielen Farben handelt, entscheide ich mich für JPG als Veröffentlichungsformat. Beim Abspeichern kann ich wählen, wie stark das Bild komprimiert werden soll. Das mittlere Bild in Abbildung 2.9 ist mit einem Faktor 50 (bei Auswahl zwischen 1 und 99) gespeichert und belegt danach noch 26 KByte. Die Bildqualität ist noch sehr gut. Das Bild ganz rechts wurde mit dem maximalen Faktor von 90 erzeugt. Es belegt zwar nur noch 4 KByte auf der Festplatte, ist aber auch von nicht akzeptabler Qualität.

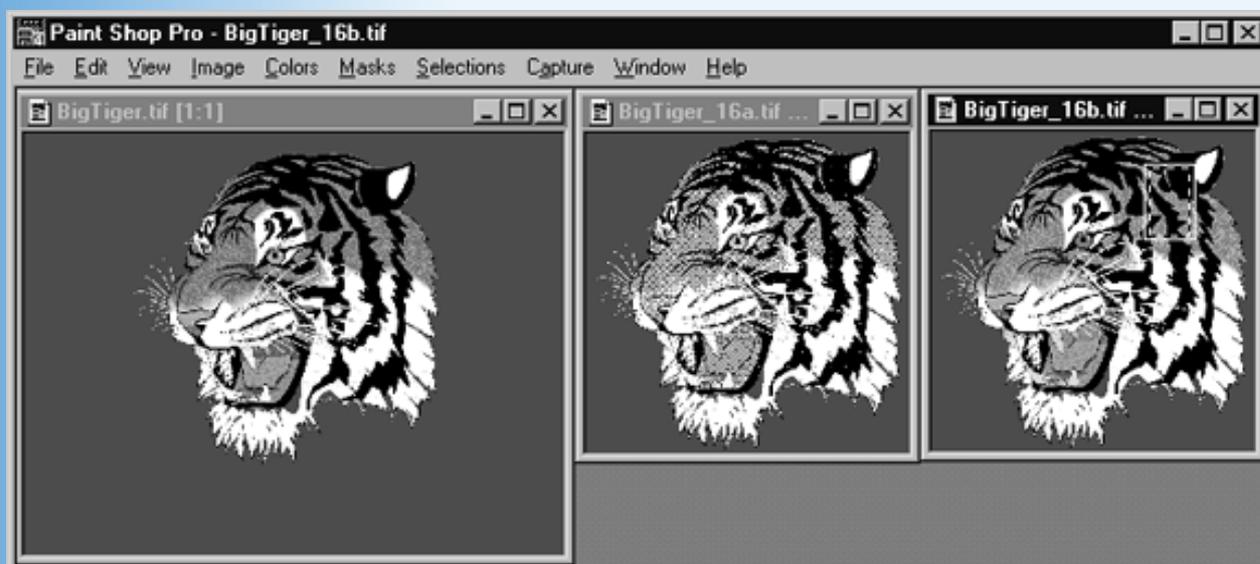


Abbildung 2.10: Reduzierung einer Grafik

Unser nächstes Bild ist eine ClipArt-Grafik mit einem Tiger-Kopf, die ursprünglich 72 KByte belegt. Der Tigerkopf soll in eine Webseite eingebaut werden, allerdings ohne den roten (sorry, die Farbe ist beim Abdruck im Buch verloren gegangen) Hintergrund. Stattdessen soll der aktuelle Hintergrund der Webseite durchscheinen.

Da wir eine transparente Farbe festlegen wollen, scheidet das JPG-Format von vornherein aus. Es würde bei der geringen Farbtiefe und der mäßigen Bildgröße vermutlich eh keine bessere Reduzierung bringen als GIF oder PNG. Doch zuerst passen wir die Bildgröße an.

Die rote Farbe um den Tiger-Kopf soll später transparent sein. Das bedeutet, das man den roten Hintergrund des Tiger-Kopfes nicht sehen wird. Also können wir den Hintergrund auf ein Minimum reduzieren. Die resultierende TIFF-Datei ist in Abbildung 2.10 nicht zu sehen, hat aber die gleichen Abmaße wie *BigTiger_16a.tif* und belegt nur noch 34 KByte.

Der nächste Schritt besteht darin, die Farbtiefe zu verringern. Die ursprüngliche Datei verwendet eine 256-Farbenpalette. Mit Paint Shop Pro können wir die Palette auf 16 Farben reduzieren, wobei uns Paint Shop Pro verschiedene Algorithmen zur Auswahl der 16 Farben zur Verfügung stellt. Der erste ausgewählte Algorithmus stellt das Rotbraun des Tigerkopfes und das Rot des Rachens nicht mehr schön dar (in der Schwarzweißdarstellung von Abbildung 2.10 sind diese Bereiche heller geworden). Im zweiten Anlauf, *BigTiger_16b.tif*, wählen wir einen Bereich aus (siehe Markierungsrahmen), dessen Farben bei der Reduzierung auf 16 Farben verstärkt gewichtet werden sollen. Dies liefert uns ein akzeptables Ergebnis.

Die fertige Datei speichern wir als non-interlaced-GIF-Datei ab (wegen der geringen Dateigröße können wir auf das Interlacing verzichten).

Zu guter Letzt legen wir noch die transparente Farbe fest. Wir lassen uns dazu die Farbpalette des Bildes anzeigen und notieren uns den Index der roten Hintergrundfarbe (die natürlich nicht für andere Teile des Bildes verwendet werden darf). Beim neuerlichen Speichern geben wir den Index der roten Farbe als Index der transparenten Farbe an. Die endgültige Datei belegt nur noch etwas mehr als 6 KByte.



Die Reduzierung der Farbtiefe vermindert die Dateigröße nicht nur dadurch, dass eine kleinere Palette abgespeichert werden muss. Hinzukommt, dass die meisten Komprimierungsverfahren um so effektiver arbeiten, je weniger Farben es im Bild gibt. Es lohnt sich also auch, die Palette von Hand zu überarbeiten und zu schauen, ob man auf einzelne Farben verzichten kann.

Bilder im Arbeitsspeicher

Als verantwortlich denkender Webautor sollten Sie nicht nur die Größe der Bilddatei, sondern auch die Größe des unkomprimierten Bildes im Arbeitsspeicher Ihrer Webbesucher im Auge behalten. Nehmen wir zum Beispiel das Foto aus Abbildung 2.9. Durch die JPG-Komprimierung konnten wir die Bilddatei von 440 KByte auf 26 KByte herunterdrücken. Trotz der stark komprimierten Bilddatei ist das eigentliche Bild aber immer noch 315*476 Pixel groß. Wird es auf dem Bildschirm eines Websurfers angezeigt, belegt jeder dieser Pixel im Speicher 24 oder 32 Bit (je nach eingestellter Farbtiefe auf dem Client-Computer).³

Noch gravierender wird der Unterschied, wenn Sie große Hintergrundbilder mit wenigen Farben verwenden. Diese lassen sich durchaus schon einmal auf Dateigrößen von unter 5 KByte reduzieren,

belegen aber im Arbeitsspeicher des Anwenders dann über 1 MByte.

2.6 Hyperlinks

Hyperlinks ermöglichen es dem Besucher einer Webseite, per Mausklick zu einer anderen Stelle der Webseite, zu einer anderen Webseite des gleichen Webs oder zu einer ganz anderen Website zu springen. Diese weltweite Vernetzung von Webseiten hat wesentlich zum Erfolg des World Wide Webs beigetragen.

Hyperlinks zu anderen Webseiten

Innerhalb des <body>-Tags haben wir es mit zwei Arten von Links zu tun:

- Links, die zu einer bestimmten Webressource führen und die über das Attribut des zugehörigen HTML-Tags definiert werden (beispielsweise Bilder, Applets oder CGI- Programme zur Bearbeitung von Formulareingaben).
- Links, die auf eine andere Textstelle oder eine andere Webseite verweisen.

Links der letzten Kategorie werden mit dem <a>-Tag eingefügt:

```
<a href="http://www.sun.com/index.html">die SUN-Website</a>
```

Ein solcher Link besteht aus drei Teilen:

- den Anker-Tags: <a>...
- dem href-Attribut, dem als Wert der URL der anzusteuernden Webseite zugewiesen wird
- einem Text, der auf der Webseite erscheint und vom Webbrowser entsprechend formatiert wird (üblicherweise spezielle Farbe und Unterstreichung).

In eine Webseite könnte ein solcher Hyperlink beispielsweise wie folgt eingebaut werden:

```
<p>Besuchen Sie doch mal <a HREF="http://www.sun.com/index.html">die SUN-Website</a> und informieren Sie sich über die neuesten Produkte!</p>
```

Hyperlinks zu anderen Websites

Um einen Hyperlink zu einer Seite eines anderen Webs zu definieren, geben Sie den vollständigen URL der Webseite an.

```
<a href="http://www.server.com/verzeichnis/">Text des Hyperlinks</a>
```

Wenn Sie einen Hyperlink zu einer anderen Website anbieten und der Besucher Ihrer Website diesen Hyperlink anklickt, verlässt er damit Ihre Website und in seinem Browser erscheint stattdessen die Webseite der anderen Site. Zwar kann der Websurfer jederzeit über den Zurück-Schalter seines Browser wieder in Ihr Web eintreten, doch für den Moment ist Ihre Webseite weg.

So etwas kann recht ärgerlich sein. Beispielsweise dann, wenn der Websurfer gerne die Inhalte der Seiten parallel lesen würde oder schnell wieder zurück auf Ihre Seite springen möchte. Wenn Sie also einen externen Hyperlink auf eine Webseite anbieten, auf der der Websurfer Zusatzinformationen zu dem Text Ihrer Webseite finden kann, und Sie davon ausgehen können, dass der Websurfer nach dem Lesen dieser Webseite auf jeden Fall wieder zu Ihrer Webseite zurückkehren möchte, vereinfachen Sie ihm doch die Navigation, indem Sie den Hyperlink so einrichten, dass die angesteuerte Webseite in einem eigenen

Browser-Fenster angezeigt wird. Sie müssen dazu nur im Ankerelement das target-Attribut verwenden und auf _blank setzen (siehe auch Kapitel 3.2.4).

```
<a href="http://www.server.com/verzeichnis/" target="_blank">Text des  
Hyperlinks</A>
```



Verwendet die Zielwebsite einen anderen Zeichensatz, können Sie potentiellen Problemen vorbeugen, indem Sie den Webbrowser auf den zu verwendenden Zeichensatz hinweisen:

```
<A href="http://www.w3.org/" charset="ISO-8859-1">W3C-Website</A>
```

Hyperlinks zu Webseiten der eigenen Website

Links zu anderen Webseiten der eigenen Website sollten grundsätzlich als relative Hyperlinks angegeben werden. Als Basis dient dabei die Adresse, die im Header-Bereich des Webdokuments mit Hilfe des <base>-Tags spezifiziert wurde oder - falls das <base>- Element nicht verwendet wurde - die URL-Adresse des aktuellen Dokuments.

Schauen wir uns hierzu ein paar Beispiele an.

```
<!doctype ...  
<html>  
<head>  
  <title>Demo</title>  
  <base href="http://www.server.com/verz/">  
</head>  
<body>  
  <p>Hier <a href="seite.html">klicken</a>  
  </p>  
</body>  
</html>
```

Der resultierende Link lautet:

```
http://www.server.com/verz/seite.html
```

Oder

```
<base href="http://www.server.com/verz/">  
...  
<a href="./Unterverzeichnis/seite.html"></a>
```

Der resultierende Link lautet:

```
http://www.server.com/verz/unterverzeichnis/seite.html
```

Oder

```
<base href="http://www.server.com/verz/">
...
<a href=" ../verzeichnis2/seite.html"></a>
```

Der resultierende Link lautet:

<http://www.server.com/verzeichnis2/seite.html>

Oder

```
<!doctype ...
<html>
<head>
  <title>Demo</title>
  <!-- keine Basisadresse -->
</head>
<body>
  <p>Hier <a href=" ./produkte/seite.html">klicken</a>
  </p>
</body>
</html>
```

Die angesteuerte Webseite steht in dem Unterverzeichnis »Produkte« des Verzeichnisses, in dem das aktuelle Dokument steht.

Hyperlinks zu bestimmten Textstellen

Wenn Sie auf einer Webseite einen Link zu einer anderen Textstelle der gleichen Webseite erstellen wollen, müssen Sie an der betreffenden Textstelle zuerst einen Zielanker setzen.

Dazu klammern Sie die anzusteuernde Textstelle in ein Ankerelement, das Sie mit einem Namen versehen:

```
<h2><a name="Zielmarke1">Abschnittsüberschrift</a></h2>
...
```

Wenn die anzusteuernde Textstelle bereits mit einem HTML-Tag beginnt, können Sie dieses zu einem Anker ausbauen, indem Sie es mit Hilfe des id-Attributs kennzeichnen.

```
<h2 id="Zielmarke1">Abschnittsüberschrift</h2>
...
```



Auf einer Webseite sollte es keine zwei HTML-Elemente mit gleichen name- oder id-Bezeichnern geben. Dagegen macht es durchaus Sinn, in einem Ankerelement sowohl name- als auch id-Attribut mit gleichen Bezeichnern anzugeben: . Der Grund hierfür ist, dass in XHTML (siehe Kapitel 21) das id-Attribut favorisiert wird, dass man derzeit aber aus Rücksicht auf ältere und aktuelle Webbrowser auch das name-Attribut noch angeben sollte.

Jetzt können Sie von überall auf diese Textstelle verweisen

- von einer anderen Textstelle aus:

```
<a href="#Zielmarkel">Zu Abschnitt 1</a>
```

- oder auch von einer anderen Webseite aus:

```
<a href="andereSeite.html#Zielmarkel">Zu Abschnitt 1 von ...</a>
```

Hyperlinks zu Textstellen kann man beispielsweise zum Aufbau von Inhaltsverzeichnissen nutzen.

Inhaltsverzeichnisse

Größere Dokumente, wie zum Beispiel die HTML-Spezifikation, die im World Wide Web veröffentlicht werden, werden üblicherweise auch mit einem Inhaltsverzeichnis ausgestattet. Dass die einzelnen Einträge im Inhaltsverzeichnis dabei mit Hyperlinks zu den betreffenden Textabschnitten versehen werden, gehört zum guten Ton.

Ein HTML-Inhaltsverzeichnis zu diesem Kapitel könnte beispielsweise wie folgt aussehen:

Listing 2.5: Auszug aus `inhaltsverzeichnis.html`

```
<h2>Inhaltsverzeichnis</h2>
<ol style="list-style-type: none">
  <li><a href="#Abschnitt1"> 1 HTML-Grundkurs</a></li>
  <ol style="list-style-type: none">
    <li><a href="#Abschnitt1.2">1.2 Das HTML-Grundgerüst</a></li>
    <li><a href="#Abschnitt1.3">1.3 Text und Überschriften</a></li>
    <li><a href="#Abschnitt1.4">1.4 Listen</a></li>
    <li><a href="#Abschnitt1.5">1.5 Bilder</a></li>
    <li><a href="#Abschnitt1.6">1.6 Hyperlinks </a></li>
    <li><a href="#Abschnitt1.7">1.7 Sonstige Tags </a></li>
    <li><a href="#Abschnitt1.8">1.8 Inline-Stile, Farben, Schriften und
      Seitenhintergrund</a></li>
  </ol>
</ol>
<h2><a name="Abschnitt1">1 HTML-Grundkurs</a></h2>
<p> Text ...</p>
<h3><a name="Abschnitt1.2">1.2 Das HTML-Grundgerüst</a></h3>
<p> Text ...</p>
...
```



Wenn Sie größere Dokumente ins Internet stellen, sollten Sie sich überlegen, ob Sie das Dokument nicht auch zum Herunterladen anbieten: in dem Sie entweder das gesamte Dokument in eine einzige HTML-Datei packen oder eine ZIP-Datei mit den einzelnen HTML-Seiten zum Download anbieten. Falls Sie dies tun, achten Sie bitte darauf, dass Sie für

Querverweise innerhalb des Dokuments nur relative URLs verwenden und keine Basisadresse angeben. So ist sichergestellt, dass der Leser den Links auch nach dem Herunterladen nachgehen kann.

Hyperlinks zu anderen Protokollen

Hyperlinks können nicht nur andere Webseiten ansteuern. Wenn Sie wollen, können Sie auch zu einem anderen Protokoll wechseln.

| Protokoll | Ziel |
|-----------|---|
| http: | HTML-Dokument |
| mailto: | E-Mail-Adresse abschicken |
| ftp: | Verweis auf eine Datei mit Hilfe des File-Transfer-Protokolls |
| news: | Verweis auf einen Newsgroup-Artikel |
| file: | Zugriff auf lokal gespeicherte Dokumente |
| gopher: | Gopher-Dokument |
| wais: | Wide Area Information System |

Tabelle 2.7: Hyperlinks und Protokolle

Bilder als Hyperlinks

Bisher haben wir nur Hyperlinks definiert, die von Textstellen ausgingen (d.h., die auf der Webseite als unterstrichener Text dargestellt werden). Es besteht aber auch die Möglichkeit, Grafiken als Hyperlinks einzurichten.

Dazu fassen Sie das ``-Tag des Bildes einfach in ein Ankerelement ein:

```
<a href="andereseite.html"></a>
```

Die Kombination Grafik/Hyperlink wird häufig zum Aufbau von Navigationsleisten genutzt (siehe Abschnitt 2.6.8). Außerdem wird dieses Verfahren eingesetzt, um Bilder- oder Fotogalerien ins Netz zu bringen. Dazu werden für die einzelnen Bilder herunterskalierte Kopien angelegt (etwa 50x70 Pixel), die übersichtlich auf einer Seite präsentiert werden. Die kleinen Bilder, die man auch Thumbnails nennt, werden in Hyperlinks eingebunden, über die der Betrachter auf Wunsch ein Bild in Originalgröße aufrufen kann.

ImageMaps - Bilder mit mehreren Links

Im einfachsten Fall ist immer genau eine Grafik mit einem Hyperlink verbunden. Es ist aber auch möglich, eine Grafik mit mehreren Hyperlinks zu verbinden. Doch wie geht das?

Natürlich funktioniert es nicht, wenn man einfach mehrere Hyperlink-Anker um eine Grafik legt und dann hofft, dass sich der Browser schon den richtigen Link aussuchen wird, wenn der Leser der Website auf die Grafik klickt.

Der Trick ist, einzelne Teilbereiche für die Grafik zu definieren und diese mit je einem Link zu verbinden. Dies geschieht mit Hilfe des <map>-Tags.⁴

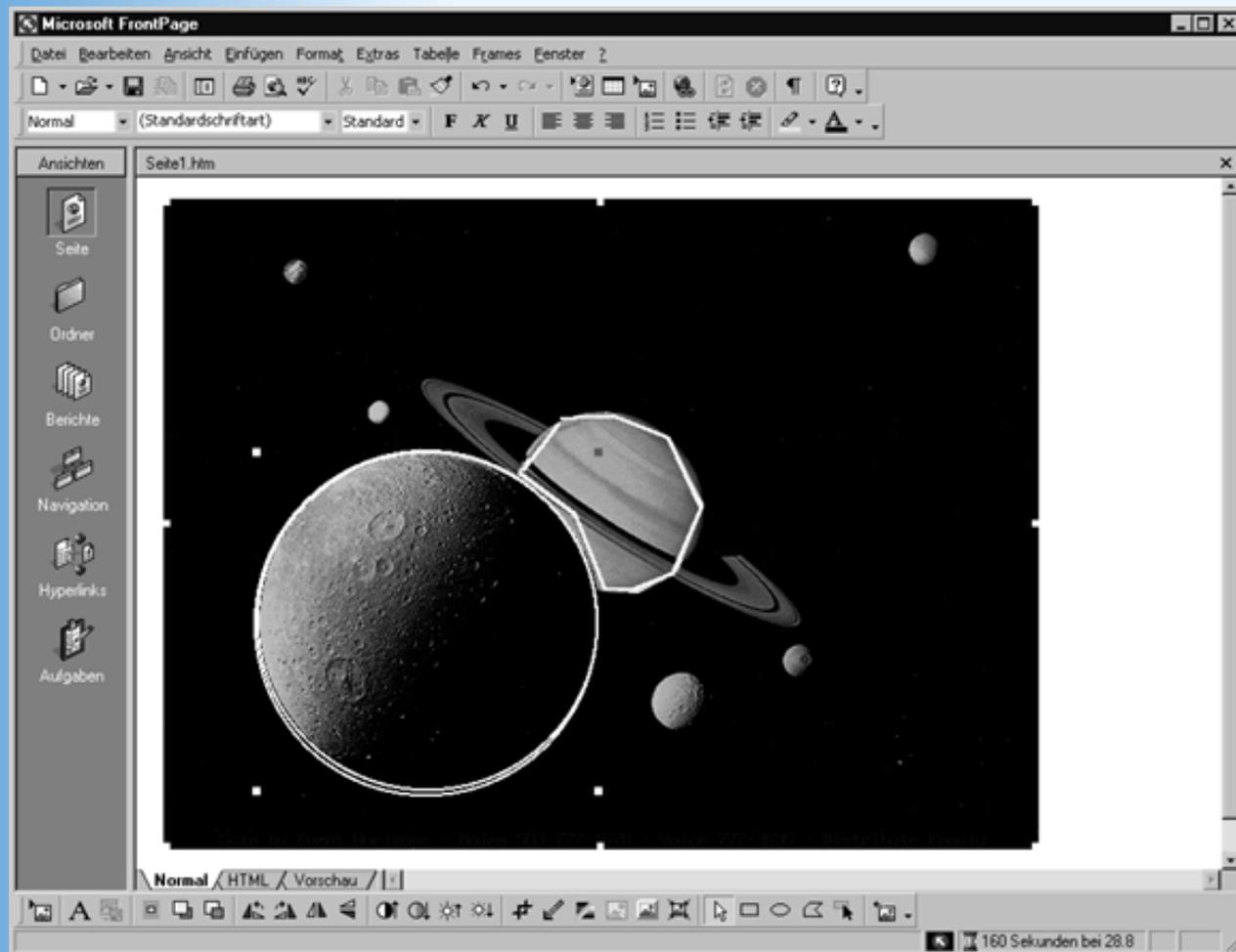


Abbildung 2.11: Erstellung einer ImageMap in einem Webeditor

Schauen Sie sich jetzt bitte einmal Abbildung 2.11 an. In dieser Abbildung sehen Sie die Grafik und die Teilbereiche, für die wir eine ImageMap erstellen wollen.

1. Als Erstes fügen wir die Grafik in die Webseite ein:

```

```

2. Dann definieren wir die <map>-Abbildung. Um die ImageMap später referenzieren zu können, weisen wir ihr mit Hilfe des name-Attributs einen Namen zu.

```
<map name="meineMap">
</map>
```

3. Die einzelnen Teilbereiche, die mit Links verbunden werden sollen, werden mit Hilfe des <area>-Tags eingefügt.

Jedes <area>-Element enthält drei Attribute:

- href zur Angabe des Ziel-URLs.
- shape, um anzugeben, welche Umrissform der Teilbereich hat
- coords zur Angabe der Eckkoordinaten, die den Umriss beschreiben.

Welche Koordinaten Sie in welcher Reihenfolge angeben, hängt von der gewählten Umrissform ab:

| shape-Typ | Koordinaten |
|-----------|--|
| default | das ganze Bild |
| rect | rechteckiger Ausschnitt

angegeben werden: die x- und die y-Koordinate der oberen linken und dann der rechten unteren Ecke. |
| circle | kreisförmiger Ausschnitt

angegeben werden: die x- und die y-Koordinate des Mittelpunktes und dann der Radius. |
| poly | polygonaler Ausschnitt

angegeben werden: die x- und die y-Koordinate der Ecken des Polygons. Wenn die letzte Ecke nicht mit der ersten Ecke zusammenfällt, schließt der Webbrowser das Polygon. |

```
<map name="meineMap" >
  <area href="s1.html" shape="circle"
    coords="193, 312, 125" />
  <area href="s2.html" shape="polygon"
    coords="260, 202, 280, 173, 324, 158, 369, 176, 394, 226,
      -375, 274, 346, 290, 321, 287, 299, 232, 300, 232" />
</map>
```



Alle Koordinaten beziehen sich auf den Ursprung (0,0-Koordinate) des Bildes. Dieser liegt in der linken, oberen Ecke des Bildes.



Geben Sie die Koordinaten in einer Zeile direkt hintereinander an (keine Umbrüche im HTML-Code).

- Schließlich verbinden Sie die Grafik mit der ImageMap, indem Sie das ``-Tag um das `usemap`-Attribut erweitern und diesem als Wert den Namen der ImageMap übergeben.

```

```

Das größte Problem bei der Erzeugung von ImageMaps ist die Angabe der Koordinaten für die anklickbaren Bereiche. Die meisten Webdesigner greifen daher zur Erstellung von ImageMaps auf spezielle Programme zurück, in denen man die Teilbereiche mit der Maus einzeichnen kann und die die

fertige ImageMap zurückliefern (siehe Abbildung 2.11). Für einfache ImageMaps kann man die Koordinaten auch schon mal von Hand eingeben, das Ergebnis im Webbrowser kontrollieren und gegebenenfalls nachbessern. Schließlich kann man sich behelfen, indem man die Grafik in ein beliebiges, pixelorientiertes Grafikprogramm lädt (beispielsweise Paint Shop Pro) und sich von dem Grafikprogramm die Koordinaten der Eckpunkte anzeigen lässt.

Serverseitige ImageMaps

Die ImageMap, die wir im vorigen Abschnitt erstellt haben, bezeichnet man auch als clientseitige ImageMap, weil die Verarbeitung der Mausklicks auf der Grafik vollständig vom Webbrowser (also auf der Clientseite) übernommen wird. Die clientseitige Verarbeitung ist schnell und hat den Vorzug, dass sie nicht den Webserver belastet. Manchmal möchte man jedoch mehr Kontrolle darüber haben, wie Mausklicks in die Grafik verarbeitet werden. In einem solchen Fall erzeugt man eine serverseitige ImageMap, die von einem speziellen Programm auf dem Webserver verarbeitet wird.

Um den Browser darauf hinzuweisen, dass ein Bild mit einer serverseitigen ImageMap verbunden ist, setzt man das Boolesche Attribut `ismap`.

```

```



Boolesche Attribute setzt man, indem man das Attribut einfach im Tag aufführt (`ismap`) oder ihm seinen Namen als Wert zuweist (`ismap="ismap"`). Wer seine HTML-Seiten XHTML-konform gestalten möchte, muss die vollständige Form verwenden (wobei zu beachten ist, dass ältere Browser, die noch nicht den HTML 4.0-Standard unterstützen, das Attribut dann unter Umständen nicht erkennen)

Im nächsten Schritt wird das ``-Tag in ein Ankerelement eingeschlossen, dessen `href`-Attribut auf das CGI-Programm verweist, das zur Bearbeitung der Mausklicks aufgerufen werden soll:

```
<a href="http://www.server.com/cgi-bin/map1">
  
</a>
```

Klickt der Anwender in die Grafik, ermittelt der Browser das verarbeitende Programm und ruft dieses auf. Dabei hängt er die Koordinaten des Mausklicks an den URL des Programmaufrufs an, beispielsweise

`http://www.server.com/cgi-bin/map1?150,270`



Wie man CGI-Programme implementiert, erfahren Sie in Kapitel 17.

Navigationsleisten

Das wohl wichtigste Einsatzgebiet für grafische Hyperlinks ist die Erstellung von Navigationselementen, mit denen man die einzelnen Seiten oder Bereiche des aktuellen Webs ansteuern kann. Schauen wir uns dazu drei Beispielseiten aus dem World Wide Web an.

Grafische Hyperlinks auf rein grafisch ausgerichteter Webseite



Abbildung 2.12: Frei verteilte grafische Hyperlinks

Die Geister und der Kürbiskopf auf dieser Seite sind mit Hyperlinks verbunden. Das Layout dieser Seite beruht übrigens auf verschachtelten Tabellen (siehe Abbildung 2.13).



Abbildung 2.13: Tabellenlayout

Navigationsleisten

Die folgende Navigationsleiste dürfte den meisten Leser von der Yahoo-Webseite bekannt sein (siehe Abbildung 2.14).



Abbildung 2.14: Navigationsleiste

Diese Navigationsleiste besteht aus einer einzigen Grafik und einer zugeordneten ImageMap:

```
<map name=t>
  <area shape=rect coords=0,0,52,52 href="http://de.yahoo.com/r/ac">
  <area shape=rect coords=53,0,121,52 href="http://de.yahoo.com/r/be">
  <area shape=rect coords=122,0,191,52 href="http://de.yahoo.com/r/bc">
  <area shape=rect coords=441,0,510,52 href="http://de.yahoo.com/r/me">
  <area shape=rect coords=511,0,579,52 href="http://de.yahoo.com/r/bd">
  <area shape=rect coords=580,0,637,52 href="http://de.yahoo.com/r/fb">
</map>

```



Die Yahoo-Seite ist schon etwas älter, daher sieht man keine Anführungszeichen um die

Attribute.

»Schaltflächen«

Ein ganz typisches Navigationselement ist auch die Auflistung von Hyperlinks zu den einzelnen (Haupt-)Seiten des Webs am linken Rand jeder Webseite. Im einfachsten Falle handelt es sich dabei um simple Text-Hyperlinks. In anspruchsvolleren, grafischen Webseiten werden grafische Hyperlinks verwendet, die sich behutsam in das Gesamtdesign der Webseite einfügen.



Abbildung 2.15: Webseite mit Navigationsschaltflächen am linken Rand

Die »Schaltflächen« aus Abbildung 2.15 bestehen aus kleinen rechteckigen Grafiken (von denen ich eine in der Abbildung weiß eingerahmt habe), die in ein Ankerelement eingefasst und auf diese Weise mit einem Hyperlink verbunden sind.

Das Hauptproblem bei der Generierung solcher Navigationselemente liegt nicht in der Technik (diese ist, wie sie in den vorangehenden Abschnitten gesehen haben, gar nicht so komplex), sondern in der Erzeugung der Grafiken. Hierfür verwendet man am besten spezielle Grafikprogramme, mit denen man Bilder glätten und zerteilen kann.

Glätten (Antialiasing)

Treffen in einer Pixelgrafik zwei Volltöne aufeinander, führt dies für gewöhnlich zur Treppenbildung (siehe Abbildung, oberer Schriftzug).



Klick mich



Klick mich

Um diese Treppenbildung zu verhindern, bedient man sich des Glättens oder Anti-Aliasings. Dabei werden die Ränder durch das Einmischen von Zwischentönen verwischt. Bessere Grafikprogramme (u.a. auch Paint Shop Pro) verfügen über Funktionen zum Glätten.

Zerteilen

Beim Aufbau komplexerer grafischer Webseiten bedient man sich häufig eines Tricks: Man erzeugt das Gesamtbild der Webseite, zerlegt dieses und teilt es auf die verschiedenen grafischen Elemente auf. Zum Schluss baut man das Bild bei der Erstellung der Webseite wieder zusammen (mit Tabellen oder durch CSS-Positionierung).

2.7 Sonstige Tags

Was für HTML-Tags gibt es sonst noch? Greifen wir in die HTML-Fundgrube und schauen wir, was zutage tritt.

Horizontale Linien

Eine beliebte Form der Gliederung für HTML-Dokumente ist die Abgrenzung von Textpassagen durch horizontale Linien mit dem `<hr />`-Tag.

Unter »echten« Webdesignern ist der Gebrauch dieses Tags allerdings verpönt.

Texte logisch strukturieren mit `div` und `span`

Mit dem HTML-Tag `<div>` können Sie mehrere HTML-Tags auf Blockebene zusammenfassen.

```
<div>
  <h2>Überschrift in &lt;div&gt;-Tags </h2>
  <p>Erster Absatz in &lt;div&gt;-Tags.</p>
  <p>Zweiter Absatz in &lt;div&gt;-Tags.</p>
  <p>Letzter Absatz in &lt;div&gt;-Tags.</p>
</div>
```

Das Pendant dazu ist das ``-Tag, mit dem Sie eine Textpassage innerhalb eines Blocks definieren können.

```
<p>Text Text Text... <span>Dieser Satz ist in &lt;span&gt;-Tags eingefasst.
</span>Dieser Satz nicht mehr.</p>
```

Worin liegt der Vorteil dieser HTML-Tags?

Mit ihrer Hilfe kann man beliebig große Textpassagen und Seitenabschnitte in HTML- Tags zusammenfassen, ohne dass damit eine Standardformatierung von Seiten des Webbrowser einhergeht. Dafür kann der Autor diese Abschnitte mit Hilfe von Stylesheets einheitlich formatieren:

```
<body style="font-family: 'Times Roman'">
  <p>Erster Absatz in Body-Abschnitt.</p>

  <div style="font-family: 'Comic Sans MS'">
    <h2>Umschalten auf Comic Sans MS</h2>
    <p>Erster Absatz in &lt;div&gt;-Tags.</p>
    <p>Zweiter Absatz in &lt;div&gt;-Tags.</p>
    <p>Letzter Absatz in &lt;div&gt;-Tags. <span style="font-family:
      Arial">Dieser Satz ist in &lt;span&gt;-Tags eingefasst.
    </span>Dieser Satz nicht mehr.</p>
  </div>
  <p>Letzter Absatz in Body-Abschnitt.</p>
</body>
```



Abbildung 2.16: Formatieren mit div und span

Zitate mit blockquote und q

HTML kennt zwei spezielle Tags zur Darstellung von Zitaten.

`<blockquote>` dient zum Kennzeichnen von Zitaten, die aus einem oder mehreren Blöcken bestehen. Der in den `<blockquote>`-Tags eingeschlossene Text wird vom Browser eingerückt (siehe Abbildung 2.4).

```
<blockquote><p>Die Haare ...</p></blockquote>
```

Will man lediglich eine Textpassage aus einem Absatz als Zitat kennzeichnen, kann man die logische Auszeichnung `<cite>` (siehe Abschnitt 2.3.4) oder das Tag `<q>` verwenden.

```
<p>Roman sagte zu mir: <q lang="en-us">Was machst du die Augen zu? Sieh doch,
er schafft es, der Verrückte!.</q></p>5
```

Laut HTML-Standard sollen die Webbrowser, den in dem `<q>`-Tag stehenden Text in Anführungszeichen klammern. Durch die Angabe der gewünschten Sprache (`lang`-Attribut) kann der Webautor vorgeben,

welche Art von Anführungszeichen («, », etc.) verwendet werden sollen. Unserer Erfahrung nach wird dies derzeit aber noch von keinem Browser unterstützt.

Stammt das Zitat von einem Webdokument, kann man sowohl für `<blockquote>` als auch für `<q>` die Adresse des Dokuments als Wert des `cite`-Attributs angeben (die aber von den meisten Browsern nicht angezeigt wird):

```
<blockquote cite="http://www.fiktiv.com/eisbaer.html"><p>Die Haare ...
```

Was gibt es sonst noch?

Auf einige HTML-Tags werden wir weder hier noch am morgigen Tag eingehen. Wenn Sie sich für eines dieser Tags interessieren, schlagen Sie es in der Original-HTML- Spezifikation (<http://www.w3.org>) nach.

| Tag | Bedeutung |
|------------------------------|--|
| <code></code> | Zur Kennzeichnung von Text, der gegenüber einer früheren Version des Webdokuments gelöscht wurde. |
| <code><ins></code> | Zur Kennzeichnung von Text, der in einer früheren Version des Webdokuments noch nicht vorhanden war. |
| <code><address></code> | Zum Einfügen von Kontaktinformationen gedacht. |

2.8 Inline-Stile, Farben, Schriften und Seitenhintergrund

Sie haben nun mittlerweile fast alle wichtigen HTML-Elemente kennen gelernt. Sie wissen jetzt, wie die HTML-Elemente aufgebaut sind, wie man sie kombiniert, wie man sie sinnvoll einsetzt. Wenig haben Sie dagegen über die Formatierung der HTML-Elemente gehört. Dies wollen wir nun nachholen.

Traditionelle Formatierung versus Stylesheets

Grundsätzlich ist zwischen der althergebrachten Formatierung mit HTML-Attributen und der Formatierung mit Stylesheets zu unterscheiden. Anhand des `<body>`-Tags wollen wir uns den Unterschied verdeutlichen.

Wie Sie wissen, umschließt das `<body>`-Tag den eigentlichen Inhalt der Webseite. Dies macht das `<body>`-Tag für den Webdesigner besonders interessant: über das `<body>`-Start- Tag kann er Formatierungen angeben, die für die gesamte Webseite gelten.

Traditionell waren dafür die Attribute `background` (Hintergrundbild), `bgcolor` (Hintergrundfarbe), `text` (Textfarbe), `link` (Farbe für Hyperlinks), `vlink` (Farbe für bereits besuchte Hyperlinks) und `alink` (Farbe für aktivierte Hyperlinks) vorgesehen.

Beispiel:

```
<body background="hintergrund.tif" text="black"
      link="red" alink="fuchsia" vlink="maroon">
...
</body>
```

Prinzipiell kann man diese Attribute auch heute noch verwenden, doch wird in der HTML 4-Spezifikation von ihrem Gebrauch abgeraten. Sie sind als »deprecated« eingestuft, was bedeutet, dass sie in zukünftigen Versionen des Standards womöglich nicht mehr berücksichtigt und in ferner, ferner Zukunft

wohl auch nicht mehr von den Browsern unterstützt werden.



Denken Sie daran, dass Sie im <DOCTYPE>-Element die Dokumententypdeklaration loose.dtd angeben, wenn Sie deprecated-Elemente verwenden.



In Anhang C finden Sie eine Liste aller - auch der »deprecated« - HTML-Tags und -Attribute.

Als Ersatz für die traditionelle Formatierung mit den deprecated HTML-Tags und -Attributen unterstützt HTML die Verwendung von Stylesheets.

Stylesheets stellen eine ausgesprochen leistungsfähige und umfangreiche Technologie dar, mit der wir uns in Kapitel 4 noch in aller Ruhe und Ausführlichkeit beschäftigen werden. Eine besonders einfache Form der Formatierung mit Stylesheets, die Verwendung von Inline-Stilen, sollen Sie aber jetzt schon kennen lernen.

Formatieren mit Inline-Stilen

Von Inline-Stilen sprechen wir, wenn wir das style-Attribut verwenden, um ein HTML- Element zu formatieren.

Der Wert des style-Attributs besteht aus einer Liste von Eigenschaft:Wert-Paaren, die voneinander durch Semikolons getrennt werden.

```
<body style="color: red; font-size:12>
```

Die wichtigsten dieser Eigenschaften sind in Tabelle 2.8 aufgelistet.

| Eigenschaft | Bedeutung |
|------------------|--|
| background-color | Hintergrundfarbe
Werte: <Farbe> transparent |
| background-image | Hintergrundbild
Werte: <uri> |
| border-color | Farbe des Rahmens um das Element.
Werte: <Farbe> transparent {1, 4} |

| | |
|-----------------|--|
| border-style | Rahmenstil

Werte: none hidden dotted dashed solid double groove ridge inset outset {1, 4} |
| border-width | Rahmenbreite

Werte: <Breite> thin middle thick {1, 4} |
| color | Vordergrundfarbe

Werte: <Farbe> |
| font-family | Schriftfamilie

Werte: Geordnete Liste von Schriftnamen |
| font-style | Schriftstil

Werte: normal italic oblique |
| font-weight | Schriftbreite

Werte: lighter normal bold bolder 100 200 300 400 500 600 700 800 900 |
| font-size | Schriftgröße

Werte: <Größenangabe in em, px ,ex, pt oder %> larger smaller |
| margin | Randbreite

Werte: <Größenangabe in em, pt, px ,ex, cm, mm oder %> {1, 4} |
| padding | Füllraum

Werte: <Größenangabe in em, pt, px ,ex, cm, mm oder %> {1, 4} |
| text-align | Textausrichtung (nur für Blockelemente)

Werte: left right center justify |
| text-decoration | Textauszeichnung

Werte: none [underline overline line-through blink] |
| text-indent | Texteinzug (nur für Blockelemente)

Werte: <Größenangabe in em, pt, px ,ex, cm, mm oder %> |
| text-shadow | Werte: none [<color> <length> <length> <length>? ,]* [<color> <length> <length> <length>?] |
| text-transform | Textumwandlungen

Werte: capitalize uppercase lowercase none |

Tabelle 2.8: Auswahl einiger allgemein gültiger Stylesheet-Eigenschaften

Die Syntax des style-Attributs und die Auswahl der Eigenschaften wird nicht von HTML, sondern durch die Stylesheet-Spezifikationen vorgegeben. HTML kann nämlich grundsätzlich mit jeder beliebigen Stylesheet-Sprache zusammenarbeiten. Die Frage ist nur, ob der Browser die Stylesheet-Informationen versteht und umsetzen kann.

Derzeit gibt es im Grunde nur eine standardisierte Stylesheet-Spezifikationen, die weltweit anerkannt und von allen aktuellen Webbrowsern unterstützt wird: die »Cascading Style Sheet«-Spezifikation, kurz CSS. Folglich ist in HTML 4.0 auch festgelegt, dass die Webbrowser grundsätzlich davon ausgehen sollen, dass im HTML-Code verwendete Stylesheets der CSS-Spezifikation folgen. Trotzdem sollte man im HTML-Dokument stets angeben, welcher Spezifikation die verwendeten Stylesheets folgen. Dazu fügt man im Header-Abschnitt der Webseite ein <meta>-Element für die Meta-Eigenschaft Content-Style-Type ein und gilt als Wert für das content-Attribut das Kürzel der Stylesheet-Spezifikation an: text/css für CSS-Stylesheets.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Meine Webseite</title>
  <meta http-equiv="Content-Style-Type" content="text/css" />
</head>
```



Wenn in diesem Buch von Stylesheets die Rede ist, dann sind immer CSS-Stylesheets gemeint.

Zum Schluss möchte ich noch einmal betonen, dass das style-Attribut nicht auf das <body>-Tag beschränkt ist, sondern auf praktisch alle HTML-Tags angewendet werden kann, die innerhalb der <body>-Tags stehen (mit Ausnahme von <param>).

Nehmen wir zum Beispiel die Stileigenschaft background-color. Wenn Sie diese Eigenschaft im <body>-Tag auf red setzen, wird der Seitenhintergrund rot eingefärbt.

Wenn Sie die Eigenschaft in einem <p>-Tag auf white setzen, erhält dieser Absatz einen weißen Hintergrund (siehe Abbildung 2.17).

```
<body style="background-color: yellow">
<p> Nehmen wir....</p>
<p style="background-color: white">Wenn Sie...</p>
</body>
```

Nehmen wir zum Beispiel die Stileigenschaft background-color. Wenn Sie diese Eigenschaft im <body>-Tag auf red setzen, wird der Seitenhintergrund rot eingefärbt.

Wenn Sie die Eigenschaft in einem <p>-Tag auf white setzen, erhält dieser Absatz einen weißen Hintergrund.

Abbildung 2.17: Hintergrundfarbe für HTML-Elemente

Allerdings gelten nicht alle Eigenschaften für alle HTML-Tags. Beispielsweise gibt es spezielle Stileigenschaften, die nur für Listen (list-style-type, list-style-image, list-style-position) oder Tabellen (table-layout, caption-side, border-collapse, empty-cells, etc.) gelten.

Andere Stileigenschaften machen nur für Blockelemente Sinn (wie <p>, <div>, <h1> bis <h6> etc.). Ein Beispiel hierfür wäre die Eigenschaft text-align mit der man die Ausrichtung eines Blockelements (linksbündig, rechtsbündig, zentriert, Blocksatz) festlegen kann. Für eine Textpassage innerhalb eines Textblocks würde diese Auszeichnung selbstverständlich keinen Sinn machen.



Einen vollständigen Überblick über die Stylesheet-Eigenschaften, ihre Werte und zugeordneten Tags finden Sie in der Datei propidx.html, die Sie sich zusammen mit der CSS2-Spezifikation von <http://www.w3.org> herunterladen können.

Farben

Über die Stileigenschaften color und background-color können Sie die Vordergrundfarbe (Textfarbe) und Hintergrundfarbe von HTML-Elementen setzen.

Die Farben werden dabei als RGB-Farben angegeben.

RGB-Farben

Die Kodierung in Rot-, Grün- und Blauanteile bezeichnet man auch als RGB-Kodierung.

Die RGB-Kodierung beruht auf dem Effekt, dass man durch Variation der Farbintensitäten für die drei Lichtfarben Rot, Grün und Blau sämtliche Farben mischen kann. Werden beispielsweise rotes, grünes und blaues Licht in voller Intensität ausgestrahlt, erhält man Weiß. Ist die Intensität aller drei Farben gleich Null (d.h., es wird kein Licht ausgestrahlt), erhält man Schwarz.

RGB-Farben werden als Kombination aus Rot-, Grün- und Blauanteil angegeben. Die Farbe Rot wäre also eine RGB-Farbe mit vollem Rotanteil ohne Grün- oder Blauanteile. In einer Stildefinition werden die einzelnen Anteile hintereinander (in der Reihenfolge Rot, Grün, Blau) als Prozentwerte (0% bis 100%), als Dezimalwerte (0 bis 255) oder als Hexadezimalwerte (0 bis ff) angegeben:

```
color: #ff0000
color: rgb(255,0,0)
color: rgb(100%, 0%, 0%)
```

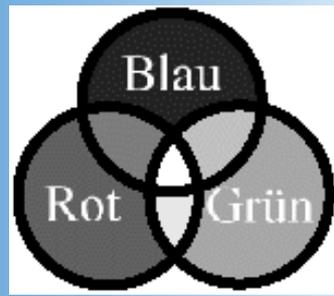


Abbildung 2.18: Das RGB-Farbmodell

Für die gängigsten Farben (genauer gesagt, die 16 Farben aus der Windows-VGA-Palette) gibt es zudem Schlüsselwörter:

| | | | |
|--------|-------|--------|---------|
| aqua | black | blue | fuchsia |
| gray | green | lime | maroon |
| navy | olive | purple | red |
| silver | teal | white | yellow |

Tabelle 2.9: Schlüsselwörter für CSS-Farben

Schriften

Zwischen Schriftensammlern und Webdesignern besteht seit jeher eine Art Hassliebe. Auf der einen Seite schätzen Designer den gezielten Einsatz von Schriften und verbringen oft Stunden und Tage damit, die für eine Webseite ideale Schrift zu finden (notfalls werden sie die Schrift selbst entwerfen), andererseits stehen Sie vor dem Problem, dass die von Ihnen ausgewählten Schriften in den Browsern Ihrer Webbesucher nicht angezeigt werden, wenn die Schriften nicht zufällig auf deren Rechner installiert sind.

Wie soll man diesem Problem entgegen treten?

- Verwenden Sie möglichst nur Schriften, die allgemein verbreitet sind (Arial, Courier New, Times Roman oder auch Helvetica, Arial Black, Comic Sans MS, Georgia, impact, Trebuchet, Verdana)
- Ersetzen Sie Elemente mit ausgefallenen Schriften durch Bilder (mit entsprechendem Schriftzug).
- Geben Sie Prioritätslisten an (siehe unten).

Wenn Sie für ein HTML-Element eine eigene Schriftart verwenden wollen, setzen Sie die Stileigenschaft `font-family`.

```
<p style="font-family: 'Times Roman' ">
```



Schriftnamen, die aus mehreren Wörtern bestehen, sollte man dabei in Anführungszeichen

setzen (einfache Anführungszeichen, da die doppelten bereits für den Stilwert verwendet werden).

Je ausgefallener die Schrift ist, um so mehr Websurfer wird es geben, auf deren Systemen die betreffende Schrift nicht installiert ist. In diesem Fall übergeht der Browser die von Ihnen vorgegebene Schrift und wählt entweder die Schrift des umliegenden HTML- Elements oder seine eigene Standardschrift. Auf diese Weise kann der Eindruck oder die Lesbarkeit einer Webseite empfindlich gestört werden.

Stellen Sie sich vor, Sie setzen in einer Webseite folgenden Satz auf:

```
<p style="font-family: 'Times New Roman'">Je ausgefallener die Schrift ist,  
<span style="font-family: 'Mein Arial'">um so mehr Websurfer wird es  
geben</span>, auf deren Systemen die betreffende Schrift nicht  
installiert ist.  
</p>
```

Die Schrift »Mein Arial« ist eine besonders ausgefallene Schrift, ähnlich Arial Black, mit der Sie den mittleren Teil des Satzes hervorheben wollen. Auf Systemen, auf denen die Schrift »Mein Arial« nicht installiert ist, wird diese Hervorhebung aber gänzlich verloren gehen, weil der Browser als Notlösung die Schrift des umliegenden HTML-Element (sprich des umliegenden Absatzes) verwendet.

Um zu verhindern, dass das Design einer Webseite gleich ganz zusammenbricht, wenn eine Schrift auf einem Clientsystem nicht vorhanden ist, kann man eine Prioritätenliste von Schriftarten angeben, beispielsweise 'Mein Arial', 'Arial Black', Arial, 'sans-serif'. Die letzte Option sollte keine direkte Schrift, sondern eine allgemeine Schriftfamilie sein. Folgende Schriftfamilien sind in CSS definiert:

| Allgemeine Schriftfamilie | Beispiel |
|---------------------------|---------------|
| serif | Times |
| sans-serif | Helvetica |
| cursive | Zapf-Chancery |
| fantasy | Western |
| monospace | Courier |

Tabelle 2.10: Allgemeine Schriftfamilien

Folgende Überarbeitung unsere obigen Beispielsabsatzes stellt sicher, dass der Browser im schlimmst anzunehmenden Fall für den mittleren Satzteil eine Schrift ohne Serifen wählt, die sich immerhin noch von der Serifen-Schrift des umliegenden Absatzes unterscheidet.

```
<p style="font-family: 'Times New Roman', serif">Je ausgefallener die Schrift  
ist, <span style="font-family: 'Mein Arial', Arial, sans-serif">um so mehr  
Websurfer wird es geben</span>, auf deren Systemen die betreffende Schrift  
nicht installiert ist.</p>
```



Außer der Schriftart können Sie noch Schriftstil (*font-style*), Schriftauszeichnung (*font-variant*), Schriftbreite (*font-weight*) und Schriftgröße (*font-size*) angeben.

Hintergrundbilder

Hintergrundbilder sind bei Laien wie bei professionellen Webdesigner recht beliebt. Während die Hintergrundbilder von Laien meist gut als solche zu erkennen sind (und leider nur zu oft viel zu aufdringlich sind), verwenden professionelle Webdesigner vorwiegend dezente Hintergründe, die meist gar nicht als solche zu erkennen sind.



Abbildung 2.19: 02bld33.pcx

Hintergrundbild

Das traditionelle Hintergrundbild ist ein rechteckiges Strukturbild von ca. 70x70 Pixeln. Kommerzielle Webeditoren stellen meist eine mehr oder weniger große Auswahl an solchen Hintergrundmotiven zur Verfügung. Wenn Sie ein solches Hintergrundbild unbedingt verwenden wollen, stellen Sie zumindest sicher, dass sich Schrift und andere Elemente Ihrer Webseite gut gegen den Hintergrund abheben.

Geben Sie den URL zu der Datei des Hintergrundbildes dann im <body>-Tag als Wert für die Stileigenschaft background-image an.

```
<body style="background-image: url('hintergrund.tif')">
```

Professionellere Seitenhintergründe erhält man, wenn man ein langes, schmales Bild nur in y-Richtung kacheln lässt.

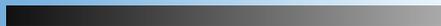


Abbildung 2.20: Bild mit Farbverlauf von Schwarz zu Blau (in Abdruck leider nur Grau)

1. Erzeugen Sie in Ihrem Grafikprogramm ein Bild von 200 x 10 Pixeln und füllen Sie es mit einem Farbverlauf von Schwarz bis Grau.
2. Wenn Sie dieses Bild als Hintergrundbild verwenden und nur in Y-Richtung kacheln lassen, erhalten Sie einen wunderschönen Streifen (siehe Abbildung 2.20).

```
<body style="background-image: url('blaustreifen.jpg');  
background-repeat: repeat-y">
```

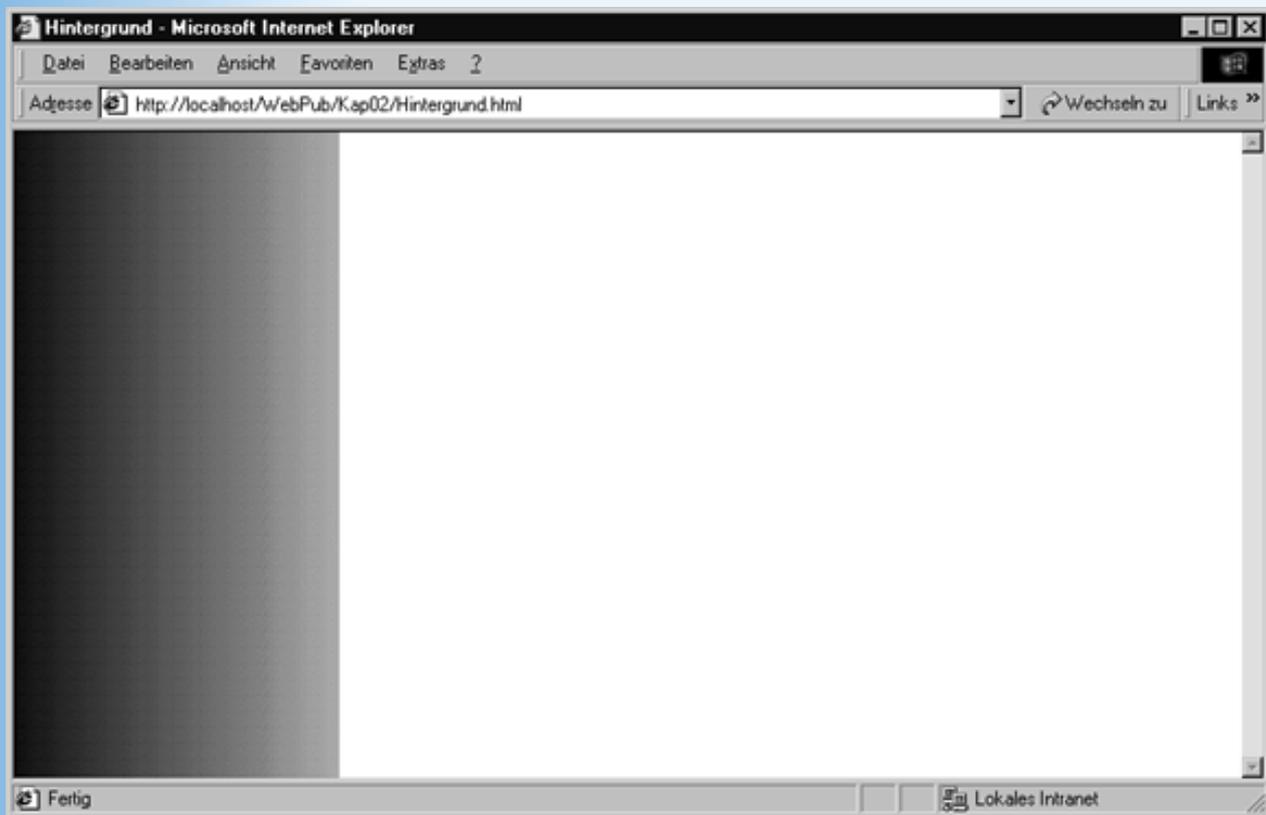


Abbildung 2.21: Streifen mit Übergang als Webhintergrund

Wenn Sie auf diesen Streifen versetzte Schaltflächen setzen (so dass diese noch ein wenig ins Weiße reichen) und den oberen Teil mit einem Logo oder Banner verzieren, erhalten Sie ein schon recht professionelles Design.



Früher, als es noch keine Unterstützung für Stylesheets gab, konnte man keinen Einfluss auf die Art der Kachelung nehmen, das heißt, die Browser legten das Hintergrundbild stets horizontal und vertikal nebeneinander aus, bis die gesamte Webseite gefüllt war. Um einen Randstreifen wie in Abbildung 2.20 zu erzeugen, darf das Bild aber natürlich nicht in X-Richtung gekachelt werden. Um dies zu verhindern, erzeugte man früher schmale Hintergrundbilder, die so lang waren, dass sie sich über die maximal anzunehmende Größe des Browserfensters erstreckten (1280 Pixel).



Prüfen Sie auch, wenn Sie ein Hintergrundbild verwenden, dass den Browserhintergrund nicht ganz ausfüllt, ob Sie die Hintergrundfarbe anpassen müssen (so dass sie der Hintergrundfarbe des Hintergrundbildes entspricht). Gehen Sie keinesfalls davon aus, dass der Hintergrund automatisch weiß ist.

2.9 Zusammenfassung

HTML (Hypertext Markup Language) ist eine Auszeichnungssprache zur Erstellung von elektronischen Dokumenten. Gemäß HTML 4.0 sollte jede HTML-Seite eine Versionsinformation, einen Header-Abschnitt mit Titelangabe und einen Body-Abschnitt mit dem eigentlichen anzuzeigenden Text enthalten.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Das HTML-Grundgerüst</title>
  </head>
  <body>
    <p>Hallo vom Browser!</p>
  </body>
</html>
```

Zur Auszeichnung und Strukturierung des Textes definiert der HTML-Standard eine Reihe von sogenannten Tags. Tags, die einen Inhalt haben, schließen diesen in ihrem Start- und End-Tag ein: `<p>Dies ist ein Absatz</p>`. Leere Tags haben kein End-Tag, dafür sollte man das Start-Tag mit einem Leerzeichen und einem Slash abschließen: `<hr />`.

In den meisten Tags können zusätzliche Optionen als Attribute angegeben werden.

```

```

Der überwiegende Teil der HTML-Tags dient der Auszeichnung spezieller Seitenelemente, beispielsweise:

- `<p>` für Absätze
- `<h1>` ... `<h6>` für Überschriften
- ``, ``, `` für Listen
- `` für Bilder
- `<var>`, `<code>`, `<cite>` für bestimmte Textstellen
- `<a>` für Hyperlinks und Textmarken

Diese Tags sind reine Struktur-Tags, das heißt, sie teilen den Webbrowsern mit, aus welchen Elementen eine Webseite aufgebaut ist, geben ansonsten aber keine weiteren Hinweise auf die Formatierung dieser Elemente. Der Webbrowser entscheidet dann, wie er eine Überschrift der 1. Ebene oder eine Variable (`<var>`) darstellt.

Webdesignern, die mehr Einfluss auf die Darstellung der Seitenelemente im Webbrowser ausüben wollen, stehen zwei unterschiedliche Wege offen:

- Sie können spezielle HTML-Tags und -Attribute verwenden, die explizite Formatanweisungen an den Webbrowser übermitteln (beispielsweise ``, `<center>` oder das `text`-Attribut für das `<body>`-Tag). Der HTML-Standard rät aber mittlerweile von der Verwendung der meisten dieser Tags und Attribute ab und empfiehlt stattdessen die Formatierung mit Stylesheets.
- Sie können Stylesheets zur Formatierung der Seitenelemente definieren. Diese können als eingebettete Stylesheets im Header-Abschnitt des HTML-Dokuments definiert werden (siehe Kapitel 3) oder in Form von Inline-Stilen über das `style`-Attribut einzelnen HTML-Tags zugewiesen werden.

Besonders hervorzuheben, sind auch die Tags `<div>` und ``, mit deren Hilfe man zusammengehörende Blöcke (`<div>`) oder einzelne Textpassagen in Blöcken (``) auszeichnen kann. Mit `<div>` kann man aufeinanderfolgende Blöcke gemeinsam formatieren, mit `` kann man eine Textpassage anders formatieren als den umliegenden Block.

2.10 Fragen und Antworten

Frage:

Kann ich auch eigene Tags definieren?

Antwort:

Nein, nicht in HTML! HTML sieht keine Wege vor, wie Sie eigene Tags definieren könnten. Tun Sie es doch, haben Sie das Problem, dass die Webbrowser nicht wissen, wie sie Ihr Tag verarbeiten sollen.

Abhilfe wird hier der Übergang von HTML über XHTML zu XML schaffen. In XML kann man eigene Tags definieren - was im Übrigen nicht nur für Webseiten interessant ist. Wenn Sie mehr über XML und XHTML erfahren möchten, lesen Sie Kapitel 21.

Frage:

Einer meiner Bekannten verwendet auf seiner Webseite eine Laufschrift, die er mit dem Tag <marquee> erzeugt hat. Kann ich dieses Tag auch verwenden?

Antwort:

Das <marquee>-Tag ist ein proprietäres Tag, das nur vom Internet Explorer unterstützt wird. In anderen Browsern wird üblicherweise nur der nicht-animierte Text der Laufschrift angezeigt. Dem <marquee>-Tag vorzuziehen ist auf jeden Fall die Implementierung der Laufschrift als Java-Applet (siehe Kapitel 16), allerdings ist dies natürlich auch aufwendiger.

```
<marquee behaviour="scroll">Dies ist eine Laufschrift</marquee>
```

Antwort:

Dem Attribut behaviour kann man einen der Werte alternate, slide oder scroll zuweisen.

Frage:

In Abschnitt 2.8.1 wird von der Verwendung der <body>-Attribute vlink und alink für besuchte und aktivierte Links abgeraten. Wie aber kann man mit Hilfe von Stylesheets eigene Farben für noch nicht besuchte, besuchte und aktivierte Hyperlinks vorgeben, wenn es doch nur ein <a>-Ankerelement für einen Hyperlink gibt?

Antwort:

Es geht. Um mit Stylesheets für Hyperlinks drei verschiedene Farben für die Zustände nicht-besucht, besucht und aktiviert festzulegen, muss man jedoch eingebettete Stylesheets definieren und sich eines Tricks, der sogenannten Pseudo-Elemente, bedienen. Wie dies geht, erfahren Sie in Kapitel 4.4.3.

Frage:

Ich habe auf meiner Webseite eine Textzeile, die ich gerne in etwas größerer, fetter Schrift formatieren möchte. Ist es okay, wenn ich die Textzeile zu diesem Zweck in <h2>-Tags einfasse?

Antwort:

Um Gottes willen, nein! Das <h2>-Tag ist ein Struktur-Tag, das dem Webbrowser mitteilen soll, dass es sich bei dem in dem Tag eingeschlossenen Text um eine Überschrift 2. Ebene handelt. Dass der Webbrowser diese Information nutzt, um die Überschrift entsprechend zu formatieren, und Sie wissen, dass diese Formatierung üblicherweise in der Zuweisung einer größeren, fetten Schrift besteht, sollte Sie nicht verleiten, das Struktur-Tag <h2> zur Formatierung zu verwenden. Wenn Ihre Textzeile also keine Überschrift der Ebene 2 ist, verzichten Sie auf das <h2>-Tag und formatieren Sie die Textzeile stattdessen mit einem Inline-Stil, beispielsweise:

<p style="font-size: 20px; font-weight: 700">Dies ist die Textzeile</p>

2.11 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

Quiz

1. Was ist der Unterschied zwischen einem leeren Tag und einem nicht-leeren Tag ohne Inhalt?
2. Wie kann man einen Zeilenumbruch erzeugen?
3. Werden <h4>-Überschriften fett formatiert?
4. Wie kann man Bilder in Webseiten einfügen?
5. Was sind ImageMaps?
6. Wie kann man einzelne HTML-Elemente mit Inline-Stilen formatieren?
7. Wie lauten die Stileigenschaften für
 - Hintergrundfarbe
 - Textfarbe (Vordergrundfarbe)
 - Einzug
 - Schriftart
 - Rahmenbreite
8. Wie kann man zwei Textabsätze und ein dazwischen gelegenes Bild zusammen einrücken?

Übungen

1. Wandeln Sie den folgenden Text in eine Webseite um.

Webseiten mit HTML und XHTML

Inhaltsverzeichnis

1. HTML
2. XHTML

1. HTML

HTML (Hypertext Markup Language) ist eine Auszeichnungssprache zur Erstellung von elektronischen Dokumenten. Zur Auszeichnung und Strukturierung des Textes definiert der HTML-Standard eine Reihe von so genannten Tags.

2. XHTML

XHTML ist der aktuelle Versuch des W3C-Konsortiums eine Brücke zwischen HTML und XML zu schlagen. XHTML-Dokumente sind einerseits den HTML-Dokumenten sehr ähnlich und können von allen gängigen Browsern korrekt angezeigt werden,

andererseits sind XHTML-Dokumenten XML-konform.

Unter <http://www.w3.org> finden Sie weitere Informationen zu HTML und XHTML.

2. Richten Sie in dem Webdokument, das Sie in Übung 1 erstellt haben, Hyperlinks vom Inhaltsverzeichnis zu den entsprechenden Textstellen ein. Verwandeln Sie auch die Webadresse des W3C-Konsortiums in einen anklickbaren Hyperlink.
3. Erstellen Sie eine einfache private Homepage mit
 - Begrüßung
 - Foto
 - Kurzem Text
 - Hyperlinks zu interessanten Webseiten
4. Legen Sie eine neue Webseite an, in die Sie das Bild einer Deutschlandkarte einbinden. Richten Sie für die Städte Berlin, München und Saarbrücken Hyperlinks ein.

❖ Kapitel Inhalt Index **SAMS** ❖ Top Kapitel ❖

1
Der Umbruch erfolgt ohne Silbentrennung. Das ist verständlich, sonst müssten die Browser ja für alle Sprachen ein passendes Wörterbuch parat haben.

2
Die genaue Darstellung des Symbols bleibt dem Webbrowser überlassen.

3
Paletten werden nur für Bilder bis maximal 256 Farben verwendet. In der Palette wird jede Farbe durch einen RGB-Wert spezifiziert (wobei je 1 Byte zur Kodierung des Rot-, Grün- und Gelbanteils zur Verfügung stehen, d.h. die Palette kann 256 Farben aus einem Spektrum von 16 Mio. Farben enthalten). Für die einzelnen Bildpunkte braucht dann nur noch ein 1-Byte großer Index in die Farbpalette abgespeichert zu werden.

4
map ist das englische Wort für Abbildung. Ein `<map>`-Element erzeugt eine Abbildung zwischen einzelnen Bereichen einer Grafik und den Hyperlinks.

5
aus »Jakob, der Lügner«

Tag 3

Tabellen, Frames und Formulare

Am gestrigen Tag hatten wir viel Stoff zu bewältigen. Heute werden es zwar weniger Themen, aber fast genau so viel Stoff sein, mit dem wir uns zu beschäftigen haben. Beginnen werden wir mit den Tabellen, die traditionell nicht nur zur Darstellung von tabellarischen Inhalten auf Webseiten, sondern auch zur Gestaltung des Seitenlayouts genutzt werden. Um die Konzeption des Seitenaufbaus geht es auch bei unserem zweiten Thema: den Frames. Frames sind bei Webdesignern ebenso beliebt wie verpönt. Warum, das werden Sie hier erfahren. Zum Schluss lassen wir den Tag mit einer Vorausschau auf die Einrichtung von Webformularen gemütlich ausklingen.

Die Themen heute:

- Tabellen in Webseiten
- Tabellen formatieren
- Tabellen und Layout
- Frames
- Frames und Layout
- Frames und Hyperlinks
- Der Mittelweg: Inline-Frames
- Formulare und Steuerelemente

3.1 Tabellen

Tabellen kommen bei der Webseitenerstellung zwei wichtige Aufgaben zu:

- zum einen kann man mit ihrer Hilfe Tabellen in Webseiten darstellen
- zum anderen gehören sie traditionell zu den Lieblingsspielzeugen der Webdesigner, die sie zur Auflockerung des Seitenaufbaus nutzen

Schauen wir uns zuerst an, wie Tabellen definiert werden.

Tabellen aufbauen

Im einfachsten Fall ist eine Tabelle ein in Zellen und Spalten aufgeteilter Block von Daten, gegebenenfalls mit einer Überschrift versehen.

Bananen: Produktion in Mio t

| | 1990 | 1999 |
|-----------|-------|--------|
| Indien | 6,734 | 10,200 |
| Brasilien | 5,506 | 5,592 |
| Ecuador | 3,055 | 4,600 |

Abbildung 3.1: Tabelle in Webbrowser

Schauen wir uns an, wie man eine solche Tabelle aufbaut.

<table></table>

Die komplette Tabellendefinition wird in die <table>-Tags eingeschlossen. Die Definition der einzelnen Zeilen und Spalten findet innerhalb dieses Bereichs statt.

Im Start-Tag der Tabelle kann man gleich festlegen, wie dick die Rahmenelemente (Attribut border) und wie breit die gesamte Tabelle (Attribut width) sein sollen.

```
<table border="1" width="550" >
</table>
```



Die Breite der Tabelle wird üblicherweise in Pixel (siehe oben) oder in Prozent des Browserfensters (beispielsweise width="100%") angegeben.

Tabellenüberschrift

Unter dem <table>-Tag können Sie eine Überschrift für die Tabelle angeben. Die Überschrift wird in das <caption>-Tag eingeschlossen.

```
<table border="1" width="550" >
  <caption>Bananen: Produktion in Mio t</caption>
```

<tr></tr> - Zeilen definieren

Danach werden die einzelnen Zeilen angelegt. Zeilen werden in die <tr>-Tags eingefasst (tr = table row = Tabellenzeile).

```
<table border="1" width="550" >
  <caption>Bananen: Produktion in Mio t</caption>
```

<tr> ...



Die Reihenfolge ist hierbei wichtig: Erst wird eine neue Zeile definiert, dann werden die Spalten für diese Zeile definiert.

<td></td> und <th></th> - Zelleninhalt

Die einzelnen Zellen (Spalten) in der Zeile werden mit den <td>-Tags definiert (td = table data = Tabellendaten). In diesen Tags können Sie beliebigen Inhalt (Text oder Bilder) erfassen.

Für Zellen, die Überschriften enthalten (üblicherweise die Zellen in der obersten Zeile und gegebenenfalls auch der linken Spalte) fasst man in <th>-Tags definiert (th = table header = Tabellenüberschriften) ein. Der Zelleninhalt wird dann von den Browsern besonders hervorgehoben (meist durch Fettdruck und automatische Zentrierung).

```
<table border="1" width="550" >
  <caption>Bananen: Produktion in Mio t</caption>
  <tr>
    <td>&nbsp;</td>
    <th>1990</th>
    <th>1999</th>
  </tr>
```

...



Alle Zeilen einer Tabelle müssen die gleiche Anzahl von Zellen enthalten.



Das Leerzeichen in leeren Zellen soll sicherstellen, dass die Tabellendarstellung von den Browsern möglichst beibehalten wird (einige Browser behandeln ganz leere Zellen recht eigenwillig).

Die vollständige HTML-Definition der Tabelle aus Abbildung 3.1 sieht wie folgt aus:

```
<table border="1" width="550" >
```



```

    <th>1990</th>
    <th>1999</th>
</tr>
<tr>
    <th>Indien</th>
    <td>6,734 </td>
    <td>10,200</td>
</tr>
<tr>
...
</table>

```

| | Bananenproduktion in Mio t | |
|------------------|----------------------------|--------|
| | 1990 | 1999 |
| Indien | 6,734 | 10,200 |
| Brasilien | 5,506 | 5,592 |
| Ecuador | 3,055 | 4,600 |

Abbildung 3.2: Tabelle mit verbundenen Zellen

<colgroup></colgroup> - Spalten gruppieren

Mit Hilfe des <colgroup>-Tags kann man Tabellen strukturieren. Diese Strukturierung sieht so aus, dass man am Anfang der Tabelle mit Hilfe von <colgroup>-Elementen angibt, aus wie vielen Spalten die Tabelle aufgebaut ist und welche dieser Spalten zu Spaltengruppen zusammengefasst werden sollen. Am Aufbau der Tabelle ändert sich dadurch nichts, oder zumindest fast nichts. Dafür hat die Strukturierung andere Vorteile:

- Am Anfang der Tabelle sind Strukturinformationen vorhanden, die der Webbrowser zum schrittweisen Aufbau der Tabelle nutzen kann (so muss der Websurfer bei umfangreichen Tabellen, beispielsweise Tabellen mit Fotos, nicht warten, bis die Tabelle und ihre Inhalte vollständig heruntergeladen wurden, bevor er etwas zu sehen bekommt).
- Der Webdesigner kann die Spaltengruppendefinitionen zur Formatierung der Tabelle nutzen.
- Mit Spaltengruppen kann man das Gitternetz der Tabelle beeinflussen.

Greifen wir noch einmal die Tabelle aus Abbildung 3.2 auf. Diese soll mit Hilfe von <colgroups> neu formatiert werden, so dass sie wie in Abbildung 3.3 aussieht.



Die im Folgenden vorgestellte Formatierung der Tabelle ist zwar vollkommen HTML-

und CSS-konform, wird aber von vielen Browsern nicht unterstützt. In Übung 3 überarbeiten wir den HTML-Code der Tabelle, so dass die Tabelle in den wichtigsten Browsern ordentlich dargestellt wird. (In Kapitel 4.6 finden Sie allgemeine Tipps und Tricks wie man Browsern, die sich mit der Unterstützung der aktuellen Standards etwas schwer tun, ein wenig nachhelfen kann.)

| | Bananenproduktion in Mio t | |
|------------------|----------------------------|--------|
| | 1990 | 1999 |
| Indien | 6,734 | 10,200 |
| Brasilien | 5,506 | 5,592 |
| Ecuador | 3,055 | 4,600 |

Abbildung 3.3: Tabelle mit Spaltengruppen

Die Tabelle enthält drei Spalten, die nun in zwei Gruppen aufgeteilt werden.

```
<table border="1" width="550">
  <colgroup></colgroup>
  <colgroup span="2"></colgroup>
```

Wie viele Spalten zu einer Spaltengruppe gehören, gibt man über das span-Attribut an. Eine Spaltengruppe, für die kein span-Attribut angegeben wird, besteht aus einer einzigen Spalte.

Jetzt können wir die einzelnen Spaltengruppen formatieren. Die erste Spalte soll 120 Pixel breit sein. Ihr Text soll rechtsbündig ausgerichtet werden, mit einem Abstand von 20 Pixeln zum Spaltenrand. Die Zelleninhalte der zweiten Spaltengruppe sollen zentriert werden:

```
<table border="1" width="550">
  <colgroup width="120" style="text-align: right;
    padding-right: 20px"></colgroup>
  <colgroup span="2" style="text-align: center"></colgroup>
```

Schließlich sollen Gitterelemente nicht mehr zwischen allen Zellen angezeigt werden, sondern nur noch zwischen den Spaltengruppen und zwischen Überschrift- und Datenzellen. Dazu setzt man im <table>-Tag das Attribut rules auf groups. Damit zwischen den Überschrift- und Datenzellen Rahmenelemente gezogen werden, müssen wir die Zeilen in einen <thead>- und einen <tbody>-Bereich aufteilen:

Listing 3.1: Bananentabelle.html

```
<table border="1" width="550" rules="groups" style="margin-left: 30px">
  <colgroup width="120" align="right"
    style="padding-right: 20px"></colgroup>
  <colgroup span="2" align="center"></colgroup>
<thead>
  <tr>
```


Tabellen können in vielfältiger Weise angepasst und formatiert werden. Einige Formatierungsmöglichkeiten kennen Sie bereits aus dem vorangehenden Abschnitt, andere werden wir Ihnen hier nun vorstellen.

Rahmenstärke

```
<table border="0">
```

Die Rahmenstärke der gesamten Tabelle lässt sich über das border-Attribut einstellen. Wenn Sie als Wert Null angeben, erzeugen Sie eine Tabelle ohne sichtbares Gitternetz, wie sie häufig zur Gestaltung des Seitenlayouts verwendet werden (siehe nachfolgender Abschnitt).

Rahmenstil

```
<table frame="void">
```

Das Attribut frame bietet seit HTML 4.0 neue Möglichkeiten für die Gestaltung des äußeren Randes der Tabelle.

| frame-Wert | Bedeutung |
|------------|-----------------------|
| void | Kein Rahmen |
| above | Rand oben |
| below | Rand unten |
| hsides | Rand oben und unten |
| vsides | Rand links und rechts |
| lhs | Rand links |
| rhs | Rand rechts |
| box | Rand an allen Seiten |
| border | Rand an allen Seiten |

Tabelle 3.1: frame-Werte

Gitterlinien

```
<table rules="none">
```

Über das Attribut rules können Sie die Umrahmung der einzelnen Zellen steuern.

| rules-Wert | Bedeutung |
|------------|-------------|
| none | Kein Rahmen |

| | |
|--------|---|
| groups | Rahmenelemente zwischen Zeilengruppen (thead, tfoot, tbody) und Spaltengruppen (colgroup) |
| rows | Rahmenelemente zwischen Zeilen |
| cols | Rahmenelemente zwischen Spalten |
| all | Rahmenelemente zwischen Zeilen und Spalten |

Tabelle 3.2: rules-Werte

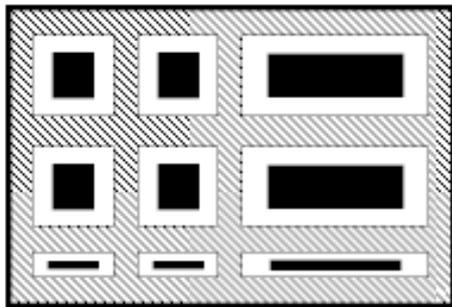
Zellenabstand

```
<table cellpadding="0">
<table cellspacing="5">
```

Über das Attribut cellpadding legen Sie den Abstand zwischen den Zelleninhalten und den Zellenrändern fest.

Über das Attribut cellspacing legen Sie den Abstand zwischen den Zellen der Tabelle fest.

Tabellenumriss ———



Cellspacing 

Cellpadding 

Zelleninhalt 

Abbildung 3.4: Spacing und Padding

Ausrichtung des Zelleninhalts

Für die Ausrichtung des Zelleninhalts gibt es verschiedene Attribute, die sowohl für die gesamte Tabelle als auch für Spaltengruppen, Zeilen oder einzelne Zellen spezifiziert werden können.

| Attribut | Bedeutung |
|----------|-----------|
| | |

| | |
|---------|--|
| align | Horizontale Ausrichtung

Mögliche Werte sind: left, center, right, justify (Block) und char (Ausrichtung an einem Zeichen) |
| valign | Vertikale Ausrichtung

Mögliche Werte sind: top, middle, bottom und baseline |
| char | Ausrichtung an einem Zeichen

Kann beispielsweise verwendet werden, um die Zahlen einer Spalte am Dezimalkomma ».« auszurichten. |
| charoff | Abstand von Ausrichtungszeichen |

Tabelle 3.3: Attribute für die Ausrichtung von Zelleninhalten

Stylesheets

Mit Hilfe von Stylesheets kann man auf jeder Ebene (von der Gesamttabelle über Spaltengruppen bis hin zu Tabellenzellen) auf die Darstellung der Tabelle und ihres Inhalts Einfluss nehmen. Im Abschnitt 3.1.1 konnten Sie bereits ein paar Beispiele zur Formatierung mit Inline-Styles verfolgen. Ausführlichere Informationen zur Formatierung mit Stylesheets finden Sie in Kapitel 4.

Webdesign mit Tabellen

Der Einsatz von Tabellen beschränkt sich nicht nur auf die tabellarische Darstellung von Daten, sondern wird auch oft zur Gestaltung des Seitenaufbaus eingesetzt. Das W3C-Konsortium rät zwar mittlerweile davon ab, Tabellen auf diese Weise zu missbrauchen, da man heute mit Hilfe von Stylesheets (CSS2 Positioning) sauberere und oft auch bessere Ergebnisse erzielen kann, doch wird die CSS2-Positionierung die Seitengestaltung mit Tabellen wohl nie ganz verdrängen können. Und dies liegt nicht nur an der Unbelehrbarkeit der Webdesigner!



Wenn Sie Tabellen zum Seitenaufbau einsetzen, nicht vergessen, den Tabellenrahmen auf 0 zu setzen: `<table border="0">`

Tabellenbreite und Seitenbreite

Die Breite einer Tabelle kann man in Pixeln oder in Prozent des Browserfensters angeben. Dies räumt uns eine Vielzahl von Gestaltungsmöglichkeiten ein.

- Wenn Sie möchten, dass Ihre Tabelle sich an die Breite des Browserfensters anpasst,

geben Sie die Tabellenbreite in Prozent an.

```
<table border="0" width="75%">
```

- Wenn Sie möchten, dass Ihre Tabelle immer das ganze Browserfenster ausfüllt, setzen Sie die Tabellenbreite auf 100 Prozent.

```
<table border="0" width="100%">
```

- Wenn Sie möchten, dass die Tabelle eine feste Breite hat, geben Sie die Tabellenbreite in Pixel an.

```
<table border="0" width="550">
```

- Wenn Sie möchten, dass die Tabelle eine feste Breite hat und zentriert im Browserfenster erscheint, geben Sie die Tabellenbreite in Pixel an. Fassen Sie die Tabelle in <div>-Tags ein und richten Sie den gesamten DIV-Block mit Hilfe des align-Attributs zentriert aus.

```
<div align="center">  
  <table border="0" width="550">  
    ...  
  </table>  
</div>
```



Die Verwendung des align-Attributs ist für das <table>-Tag eigentlich »deprecated«. Laut HTML-Standard sollte man das Attribut nur noch zur Ausrichtung des Inhalts einzelner Tabellenzellen (-zeilen oder -spalten) verwenden. Im Internet Explorer und Navigator 4 kann man Tabellen zentrieren, indem man sie in <div>-Blöcke einfasst, für die die Stileigenschaft text-align gesetzt ist: <div style="text-align: center">. Der Netscape 6-Browser interpretiert diese Formatierungsanweisung allerdings so, dass er den Inhalt der Tabelle und nicht die Tabelle selbst zentriert.

- Wenn Sie möchten, dass eine Webseite nur in begrenztem Maße an die Breite des Browserfensters angepasst wird, versuchen Sie es mit folgendem dreispaltigen Tabellenaufbau.



Abbildung 3.5: Variabler, dreispaltiger Tabellenaufbau

Dieses Layout besteht aus:

- Einer Tabelle, die die gesamte Breite des Browserfensters einnimmt (Breite = 100%).
- In der Mitte befindet sich die Spalte mit dem eigentlichen Seiteninhalt. Diese Spalte ist auf eine »feste« Breite von 450 Pixeln gesetzt.
- Links und rechts befinden sich zwei »leere« Spalten, die jeweils 10% einnehmen (Sie könnten auch nur die die linke Spalte auf 10 Prozent der Gesamtbreite der Tabelle setzen und die rechte die Restbreite absorbieren lassen (`width=""`), doch liefert dies in Netscape Navigator 4 schlechtere Ergebnisse.)
- Einem oberen und linken Rand: `<body style="margin-top: 30px; margin-left: 30px">`

Listing 3.2: segenDerErde.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Segen der Erde</title>
</head>
<body style="margin-top: 30px; margin-left: 30px;
  -background-color: black; color: white"
  text="white">
<table border="1" width="100%" cellspacing="0" cellpadding="0">
  <tr>
    <td width="10%">&nbsp;</td>
    <td width="450">
      <h1>Segen der Erde</h1>
      <p>Der lange, lange Pfad über das Moor in den Wald hinein - wer
        hat ihn ausgetreten? ...</p>
    </td>
    <td width="10%">&nbsp;</td>
  </tr>
</table>
</body>
</html>
```

Wie wird diese Webseite im Browser angezeigt?

Bei normaler Browsergröße erscheint der Text zentriert, die Stylesheet-Eigenschaft `margin-top` im `<body>`-Tag sorgt für den Abstand zum oberen Rand, die beiden leeren Tabellenspalten und die Stileigenschaft `margin-left` für den Abstand zu den seitlichen Rändern.

Betrachtet der Besucher die Webseite bei größerer Bildschirmauflösung oder in einem größerem Browserfenster, passt die linke Tabellenspalte den Abstand vom linken Seitenrand an. Der Text wird nicht weiter ausgedehnt - was für den Leser meist recht angenehm ist, denn zu lang gezogene Textzeilen, bei denen das Auge weit nach rechts und links wandern müssen, machen das Lesen anstrengend.

Betrachtet der Besucher die Webseite bei kleinerer Bildschirmauflösung oder in einem kleineren Browserfenster, verhindert der linke feste Rand, dass der Text gleich ganz an den Seitenrand gedrückt wird. Bei etwas breiterem Browserfenster wird die linke Tabellenspalte immer breiter und rückt den Text der Tabelle weiter in die Fenstermitte.

Zeitungslayout

Zeitungen und Zeitschriften zeichnen sich meist durch mehrspaltigen Text aus. Mit Hilfe einer Tabelle kann man einen solchen Seitenaufbau auf Webseiten nachstellen.

Bildergalerien

Wie soll man Bilder- oder Fotosammlungen im Web ausstellen. Eine Möglichkeit ist, jedes Bild auf einer eigenen Seite zu präsentieren. In so einem Fall wird man das Bild vermutlich mittig zentrieren. Was aber macht man, wenn man mehrere Bilder untereinander auf einer Seite präsentieren will?

Am Einfachsten ist es natürlich, die Bilder von oben nach unten aufzuführen und unter jedem Bild den beschreibenden Text einzufügen, bevor unter diesem das nächste Bild folgt. Ein solcher Aufbau ist allerdings auch recht langweilig. Spannender ist zweifelsohne eine dreispaltige, zentrierte Tabelle, in deren mittleren Spalte untereinander die Bilder aufgeführt werden, während die Beschriftungen abwechseln links und rechts stehen. Dazu ein schwarzer Hintergrund (und weiße Schrift) und die Präsentation wirkt gleich etwas edler.

Freies Layout

Dank der vielfältigen Möglichkeiten zum Aufbau von Tabellen (Zellen verbinden, Spalten gruppieren, Tabellen ineinander verschachteln), kann man mit Tabellen auch ganz freie und eigenwillige Seitenaufteilungen erzeugen. So lassen sich Textelemente und Bilder scheinbar frei auf der Webseite platzieren.



Abbildung 3.6: Freies Seitenlayout mit Tabellen

3.2 Frames

Frames dienen dazu, ein Webdokument - oder aus Perspektive des Lesers: den Anzeigebereich im Browser - aufzuteilen. Wozu kann man dieses Konzept verwenden und wie funktioniert es?

Framesets, Frames und Webseiten

Frameseiten werden in zwei Schritten erstellt:

1. Zuerst setzt man die eigentliche Frameseite auf. Diese enthält keinen anzuzeigenden Text, sondern definiert lediglich die Aufteilung des Browserfensters in Frames. Diese Seite wird der Websurfer später aufrufen.
2. Dann erzeugt man ganz normale Webseiten, deren Inhalt in den Frames angezeigt werden soll. Im Code der Frameseite kann man festlegen, welche Webseite in welchen Frame eingeblendet werden soll.

Schauen wir uns einfach mal ein Beispiel an:

Listing 3.3: zweiframes.html - Frameseite für zwei nebeneinander liegenden Frames

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
    "http://www.w3.org/TR/html4/frameset.dtd">
<html>
```

```

<head>
  <title>Frameseite</title>
</head>
<frameset cols="20%,80%">
  <frame name="inhalt" src="inhaltsverzeichnis.html" />
  <frame name="hauptframe" src="hauptseite.html" />
</frameset>
</html>

```

Was an diesem HTML-Code sofort auffällt, ist, dass es keinen `<body>`-Abschnitt gibt. Denkt man ein wenig darüber nach, wird einem schnell klar warum. Da Frameseiten keinen wirklichen anzuzeigenden Inhalt enthalten, brauchen sie auch keinen `<body>`-Abschnitt. Dafür enthalten Frameseiten einen `<frameset>`-Abschnitt, in dem die Aufteilung des Browserfensters in Frames festgelegt wird.



Die in einem `<frameset>`-Abschnitt definierte Gruppe von Frames bezeichnet man als »Frameset«.

Noch im `<frameset>`-Starttag teilen wir dem Browser mit, in wie viele Spalten (cols) das Browserfenster aufzuteilen ist. Statt der genauen Anzahl geben wir für jede Spalte eine Größenangabe an (der Browser kann dann selbst zusammenzählen, wie viele Spalten es sind).

```

<frameset cols="20%,80%">

```

Im `<frameset>`-Abschnitt werden dann die Frames für die »Spalten« definiert.

```

  <frame name="inhalt" src="inhaltsverzeichnis.html" />
  <frame name="hauptframe" src="hauptseite.html" />
</frameset>

```

Hier werden zwei Frames definiert. Der erste Frame, der 20% der Breite des Browserfenster einnehmen soll, erhält den Namen `inhalt` und wird mit dem HTML-Dokument *inhaltsverzeichnis.html* verbunden. Der erste Frame, der 80% der Breite des Browserfenster einnehmen soll, bekommt den Namen `hauptframe` zugewiesen und wird mit dem HTML-Dokument *hauptseite.html* verbunden.

Wenn Sie diese Frameseite in Ihren Browser laden (der hoffentlich Frames unterstützt), werden Sie eventuell zuerst zwei Fehlermeldungen erhalten, die Sie darauf hinweisen, dass die Seiten *inhaltsverzeichnis.html* und *hauptseite.html* nicht gefunden werden können, bevor dann schließlich doch noch die in Frames aufgeteilte Frameseite erscheint (siehe Abbildung 3.7).

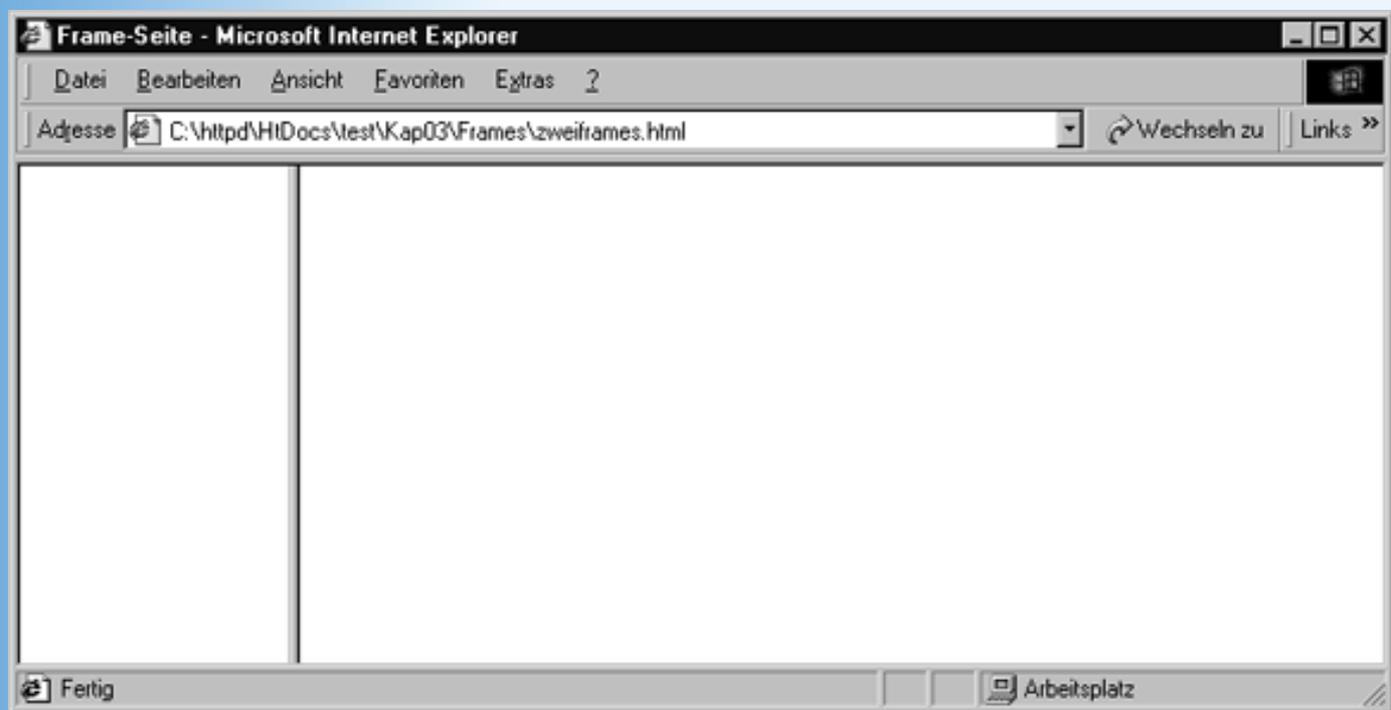


Abbildung 3.7: Frames im Browser



Wenn Sie die Frameseite von Ihrem lokalen Server aus laden, wird dieser vermutlich in die Frames entsprechende Fehlermeldungen einblenden, dass die angeforderten URLs für die Frames nicht verfügbar sind.

Seiten für die Frames aufsetzen

Um das Beispiel abzuschließen, setzen wir jetzt noch zwei einfache Seiten für die beiden Frames unserer Frameseite auf.

Für den linken Frame erstellen wir ein kleines Inhaltsverzeichnis für drei Rilke-Gedichte.

Listing 3.4: inhaltsverzeichnis.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Inhaltsverzeichnis</title>
</head>
<body>
<dl>
  <dt>Rilke</dt>
  <dd>Der Panther</dd>
  <dd>Herbsttag</dd>
```

```
<dd>Im Saal</dd>
</dl>
</body>
</html>
```

Ich habe das Inhaltsverzeichnis mit einer Definitionsliste erzeugt (siehe Kapitel 2.4). Eine Nummerierung schien mir für einen Gedichtband unangebracht und die Definitionsliste hat den Vorteil, dass die Gedichte unter dem Dichter gleich schön eingerückt werden.

In der HTML-Seite für den zweiten Frame führen wir untereinander die Gedichttexte auf.

Listing 3.5: hauptseite.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Gedichte</title>
</head>
<body>
<h2>DER PANTHER</h2>
<pre>
  Sein Blick ist vom Vorübergehn der Stäbe
  so müd geworden, daß er nichts mehr hält.
  Ihm ist, als ob es tausend Stäbe gäbe
  und hinter tausend Stäben keine Welt.
  Der weiche Gang geschmeidig starker Schritte,
  der sich im allerkleinsten Kreise dreht,
  ist wie ein Tanz von Kraft um eine Mitte,
  in der betäubt ein großer Wille steht.
  Nur manchmal schiebt der Vorhang der Pupille
  sich lautlos auf - dann geht ein Bild hinein,
  geht durch der Glieder angespannte Stille -
  und hört im Innern auf zu sein.
</pre>
<br />
<br />
<h2>HERBSTTAG</h2>
...
```

Ich verwende hier das `<pre>`-Tag, damit der Browser die Formatierung im HTML-Code (insbesondere die Zeilenumbrüche und das Einrücken mit Leerzeichen) übernimmt und ich nicht lauter `
`-Tags und ` `-Sonderzeichen einfügen muss.

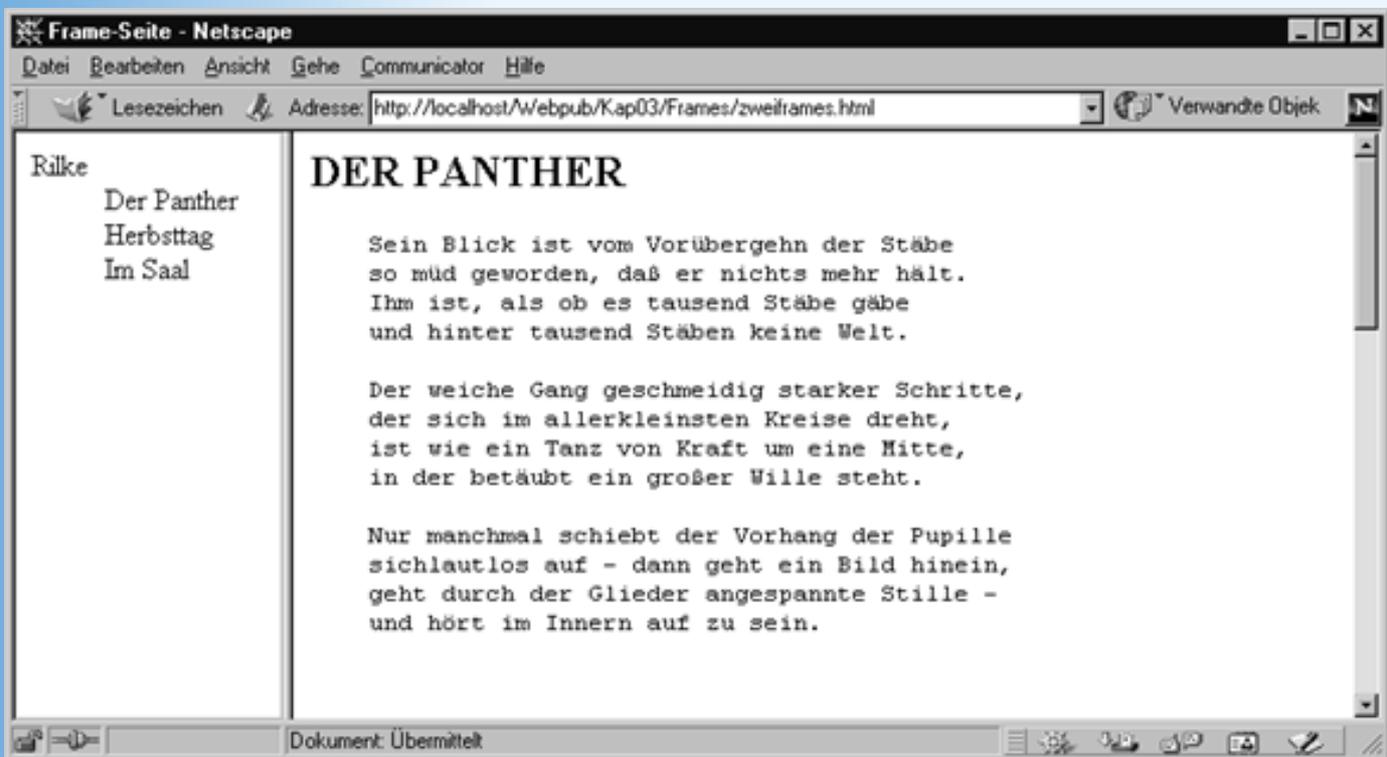


Abbildung 3.8: Frame-Aufteilung in Inhaltsverzeichnis und Textseite

Soweit wie wir es hier sehen, ist das Beispiel noch in mancher Hinsicht unvollkommen. Beispielsweise wird der Verzeichniseintrag »Der Panther« umgebrochen, wenn das Browserfenster kleiner gemacht wird. Unbefriedigend ist auch, dass man die Gedichte nicht durch Anklicken im Inhaltsverzeichnis (linker Frame) in den Anzeigebereich des rechten Frames rücken kann. Um diese Mängel zu beheben, müssen wir ein bisschen mehr über den Aufbau und die Konfiguration von Frameseiten lernen.

Framesets aufbauen

Die Aufteilung des Browserfensters in Frames geschieht wie gesagt in der Frameseite, die zu diesem Zweck statt des <body>-Abschnitts einen <frameset>-Abschnitt definiert.



Wenn Sie mit <!DOCTYPE>-Versionsinformationen in die Seite aufnehmen (was Sie tun sollten), denken Sie daran als HTML-Version HTML 4.01 Frameset//EN und als Dokumententypdeklaration frameset.dtd anzugeben (siehe Kapitel 2.2.1).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
    "http://www.w3.org/TR/html4/frameset.dtd" >
<html>
<head>
    <title>Frameseite</title>
</head>
```

```

<frameset cols="20%,80%">
  <frame name="inhalt" src="inhaltsverzeichnis.html" />
  <frame name="hauptframe" src="hauptseite.html" />

</frameset>
</html>

```

<frameset></frameset>

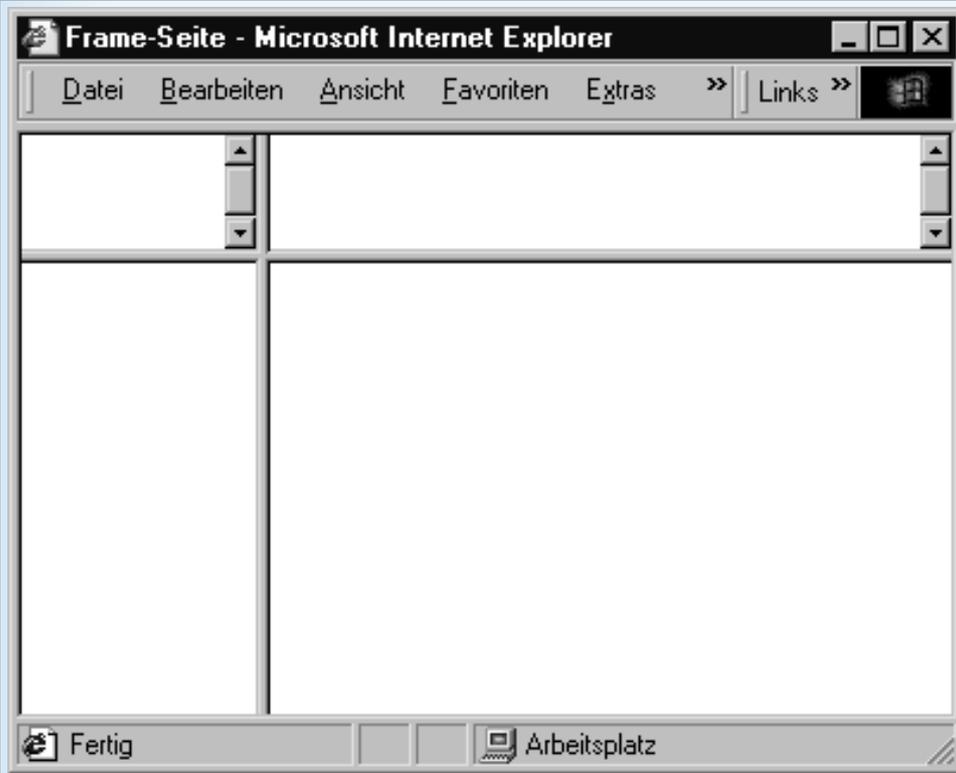
Die gesamte Frames-Deklaration wird in den <frameset>-Tags eingeschlossen. Über die Attribute rows und cols kann man festlegen, in wie viele Zeilen (rows) und Spalten (cols) das Browserfenster aufgeteilt werden soll. Wenn Sie eines dieser Attribute auslassen, geht der Browser davon aus, dass Sie nur eine Zeile oder Spalte (je nachdem, ob Sie rows oder cols weglassen) wünschen.

Als Wert für rows und cols führen Sie eine durch Kommata getrennte Liste von Größenangaben auf. Als Größenangaben kommen Pixelwerte (30, 250), Prozentangaben (30%, 75%) oder Verhältnswerte (*, 3*) in Frage. Frames mit Verhältnswerten teilen sich den Platz, der nach Abzug der Pixel- und Prozentwerte verbleibt, gemäß ihrer Verhältnswerte auf (wobei * für 1* steht).

Das klingt alles ein wenig kompliziert, wird aber schnell klar, wenn man sich ein paar Beispiele anschaut.

HTML-Code	Frames
<frameset cols="20%,80%">	Eine Zeile mit zwei Spalten, d.h. zwei nebeneinander liegende Frames. Der erste (linke) Frame nimmt 20 % des Browserfensters ein, der zweite (rechte) Frame nimmt 80 % ein.
<frameset rows="20%,80%">	Eine Spalte mit zwei Zeilen, d.h. zwei untereinander liegende Frames. Der erste (obere) Frame nimmt 20 % des Browserfensters ein, der zweite (untere) Frame nimmt 80 % ein.
<frameset cols="50,300">	Eine unsinnige Angabe! Wenn Sie für alle Frames in der Breite oder Höhe feste Werte vorgeben, gibt es keine Möglichkeit für den Browser, die Anzeige an die Abmaße des Browserfensters anzupassen. Er müsste eigentlich die Abmaße seines eigenen Fensters korrigieren. Da er dies nicht kann (oder nicht will), übergeht er Ihre Anweisungen.

```
<frameset rows="50,*"
cols="100,*">
```



Vier Frames in zwei Zeilen und Spalten.

Der erste Frame ist 50 Pixel hoch und 100 Pixel breit (erste Spalte in erster Zeile).

Der zweite Frame ist 50 Pixel hoch und nimmt in der Breite den Rest des Browserfensters ein (zweite Spalte in erster Zeile).

Der dritte Frame nimmt in der Höhe den Rest des Browserfensters ein und ist 100 Pixel breit.

Der vierte Frame nimmt in Breite und Höhe den verbleibenden Raum ein.

```
<frameset
rows="100,1*,3*">
```

Drei untereinanderliegende Frames. Der erste Frame ist genau 100 Pixel hoch. Die beiden anderen Frames teilen sich die verbleibende Höhe im Verhältnis 1:3.

Wenn das Browserfenster beispielsweise 500 Pixel hoch ist, nimmt der erste Frame davon 100 Pixel ein. Die restlichen 400 Pixel werden im Verhältnis 1:3 aufgeteilt, also 100 Pixel Höhe für den zweiten und 300 Pixel Höhe für den dritten Frame.

Tabelle 3.4: Breiten und Höhen der Frames

Frames definieren und konfigurieren

Innerhalb der `<frameset>`-Tags werden die einzelnen Frames definiert, und zwar in der Reihenfolge, in der sie auf die Zeilen und Spalten der Frameseite aufgeteilt werden sollen. Zuerst

kommen die Frames der ersten Zeile, angegeben von links nach rechts. Dann die Frames der zweiten Zeile, von links nach rechts, und so weiter bis zur letzten Zeile.

Um einzelne Frames zu definieren, verwendet man das <frame>-Tag.

```
<frame src="seite1.html" />  
<frame src="bild1.gif" />
```

Das wichtigste Attribut zu dem <frame>-Tag ist das src-Attribut, das als Wert den absoluten oder relativen URL (siehe Kapitel 2.6) zu der Webseite oder dem Bild enthält, das in dem Frame angezeigt werden soll. Doch auch die anderen Attribute sind recht interessant:

Rahmenelemente ein- und ausblenden

```
<frame frameborder="1" ...
```

Der Wert 1 zeigt dem Browser an, dass um den Frame Rahmenelemente gezeichnet werden sollen.

Der Wert 0 zeigt dem Browser an, dass um den Frame keine Rahmenelemente gezeichnet werden sollen. Das heißt aber noch nicht, dass um den Frame wirklich keine Rahmenelemente zu sehen sind. Grenzt der Frame an Frames, für die frameborder="1" ist, wird zwischen diesen Frames ein Rahmenelement eingezeichnet.



Wenn Sie für Ihre Frames das Attribut frameborder auf 0 setzen, werden zwar keine Rahmenelemente zwischen den Frames gezeichnet, doch bedeutet dies nicht, dass die Inhalte der Frames dann nahtlos ineinander übergehen. Wenn Sie beispielsweise für alle Webseiten der Frames die Hintergrundfarbe auf Rot (style="background-color: red") setzen, werden Sie feststellen, dass ein Netz weißer Linien zurückbleibt. Um diese Lücken zwischen den Frames zu schließen, müssen Sie für den Internet Explorer und den Netscape Navigator im <frameset>-Tag das Attribut border auf 0 setzen. (Hinweis: Offiziell ist das border-Attribut für das <frameset>-Tag nicht definiert.)

Rahmenelemente verschieben

```
<frame noresize="noresize" ...
```

Wenn Sie es nicht anders vorgeben, kann der Besucher Ihrer Website die Rahmenelemente zwischen den Frames mit der Maus aufnehmen und verschieben. Wenn Sie dieses Verhalten unterbinden wollen, setzen Sie in dem betreffenden Frame das Boolesche Attribut noresize.



Aber Achtung! Diese Einstellung bedeutet nur, dass die Rahmenelemente des Frames bei der Anzeige in einem Browser nicht mit der Maus verschoben werden können. Es bedeutet nicht, dass sich die Größe des Frames nicht ändert, wenn der Anwender den ganzen Anzeigebereich des Browser vergrößert oder verkleinert. Dazu müssen Sie die Größe des Frames zusätzlich in Pixel definieren. (Eine Garantie ist dies aber auch nicht, da die Frames ja zusammen den Anzeigebereich des Browsers ausfüllen müssen.)

Und nochmals Achtung! Wird die Größe eines Frames als nicht veränderbar eingestellt, bedeutet dies, dass seine Rahmenelemente nicht im Browser verschoben werden können. Dies betrifft dann aber natürlich auch die angrenzenden Frames.

Abstand von den Rahmenelementen

```
<frame marginwidth="20px" marginheight="20px" ...
```

Diese Attribute werden derzeit auf sehr seltsame Weise von den Browsern unterstützt, so dass sich ihr Effekt nur sehr schwierig voraussagen lässt. Wir empfehlen daher zum einrücken der Frame-Inhalte in den Webseiten der Frames die margin-Stileigenschaften für das <body>-Tag zu setzen: <body style="margin-left: 25px; margin-top: 25px">. Dies funktioniert aber natürlich nur für Webseiten, die man selbst aufsetzt.

Die Attribute marginwidth und marginheight sollten Sie stattdessen beide auf 0 setzen. Denn nur so können Sie sicherstellen, dass die Webseiten, die in einen Frame geladen werden, diesen auch im Netscape Navigator voll ausfüllen.

Bildlaufleisten ein- und ausblenden

```
<frame scrolling="no" ...
```

Kann in einem Frame nicht der komplette Inhalt der eingeblendeten Webseite angezeigt werden, so fügt der Browser automatisch Bildlaufleisten hinzu.

Mit dem Attribut scrolling können Sie vorgeben, ob Bildlaufleisten in Frames eingeblendet werden sollen.

Wert	Beschreibung
scrolling = "no"	Es werden keine Bildlaufleisten angezeigt.
scrolling = "yes"	Es werden immer Bildlaufleisten angezeigt.

scrolling = "auto"	Bildlaufleisten werden eingeblendet, wenn der Inhalt der Webseite nicht vollständig im Frame angezeigt werden kann.
--------------------	---

Tabelle 3.5: Bildlaufleisten in Frames

Jetzt fehlt uns nur noch ein frame-Attribut: name. Dieses Attribut ist besonders in Verbindung mit der Verwendung von Hyperlinks in Frames interessant.

Frames und Hyperlinks - ein Kapitel für sich

Für einen kurzen Moment wollen wir nun die Frameseite verlassen und wenden uns den Webseiten zu, die in den Frames angezeigt werden.

Wenn man innerhalb eines Frames auf einen Hyperlink klickt, folgt der Browser dem Link, beschafft die Webseite (oder Grafikdatei), auf die der Link verweist, und zeigt ihren Inhalt in dem Frame an, in dem der Hyperlink stand. Das bedeutet, die Seite, die aktuell in dem Frame angezeigt wird, wird durch eine neue Seite ersetzt.

Dies ist nicht immer das Verhalten, das man sich als Webautor wünscht. Am Ende des Abschnitts 3.2.1 haben wir bereits festgestellt, dass es nett wäre, wenn man über einen Hyperlink aus Frame A (in unserem Beispiel der Frame mit dem Inhaltsverzeichnis zu den Rilke-Gedichten) einen Textpassage in der Webseite eines anderen Frames ansteuern beziehungsweise eine neue Webseite in diesen Frame laden könnte. Erfreulicherweise stellt dies kein großes Problem dar.

1. Weisen Sie dem Frame, der von dem Hyperlink angesteuert werden soll, in seiner Definition im HTML-Code der Frameseite einen Namen zu:

```
<frame name="frame2" src="seite2.html" />
```

2. Geben Sie in den <a>-Ankern der Hyperlinks den Namen des Zielframes als Wert für das target-Attribut an:

```
<a href="seite_2_2.html" target="frame2" />
```

Wie wäre es mit einer kleinen Zwischenübung?

Überarbeiten Sie die Dateien *zweiframes.html*, *inhaltsverzeichnis.html* und *hauptseite.html* (Listing 3.3 bis Listing 3.5) so, dass im linken Frame Hyperlinks angezeigt werden, über die man zu den Anfängen der drei Gedichte springen kann. Als Ausgangsbasis können Sie die Listings auf der Buch-CD nehmen. Hinweis: Da alle Gedichte in einer Datei stehen, müssen Sie die Überschriften der Gedichte mit benannten Anker-Elementen versehen, die Sie ansteuern können (siehe Kapitel 2.6).

Wenn Sie irgendwo hängen bleiben, lesen Sie in Übung 4 nach, wie es geht.



Abbildung 3.9: Frameseite nach Klick auf den Herbsttag-Link

Zielframe für eine ganze Webseite

Häufig haben alle oder zumindest ein großer Teil einer Webseite den gleichen Zielframe. Im Falle unserer Gedichtesammlung haben beispielsweise alle Hyperlinks der Seite *inhaltsverzeichnis.html* den rechten Frame als Ziel.

In so einem Falle kann man sich viel Arbeit sparen, indem man den Target-Frame mit Hilfe des `<base>`-Tags im Header-Abschnitt der Webseite definiert:

```
<head>
  ...
  <base target="frame2" />
</head>
```

Was aber macht man, wenn es auf der Seite einen oder zwei Hyperlinks gibt, die einen anderen Zielframe haben?

Dann behält man den im `<base>`-Tag spezifizierten Zielframe als Standard-Zielframe der Webseite bei und gibt für alle Hyperlinks mit abweichendem Zielframe im Anker-Element ein passendes `target`-Attribut an:

```
<a href="seite.html" target="frame4">Klick mich</a>
```

Für alle Hyperlinks ohne eigenes `target`-Attribut im Anker-Element gilt dann der Zielframe aus

dem <base>-Tag. Für Hyperlinks mit target-Attribut im Anker-Element hat dieses Vorrang.

Standardnamen

Als »target« können Sie neben den von Ihnen definierten Frame-Namen auch vier von HTML definierte Standardnamen angeben.

target-Wert	Beschreibung
_blank	Das Dokument, auf das der Link verweist, wird in einem ganz neuen Browserfenster angezeigt.
_self	Das Dokument, auf das der Link verweist, wird in dem aktuellen Frame angezeigt. Dieser target-Wert ist interessant für Webseiten, die in ihrem Header-Abschnitt einen anderen Frame als Standard-Zielframe definieren (<base>-Tag).
_parent	Das Dokument, auf das der Link verweist, wird in den übergeordneten Frame (siehe nachfolgender Abschnitt zur Verschachtelung von Framesets) geladen. Wenn kein übergeordneter Frame existiert, wird das Dokument in den aktuellen Frame geladen.
_top	Das Dokument, auf das der Link verweist, wird in das aktuelle Browserfenster geladen und ersetzt dort die Frameseite. Dieser target-Wert stellt eine Möglichkeit dar, die Frame-Aufteilung des Browserfensters aufzuheben.

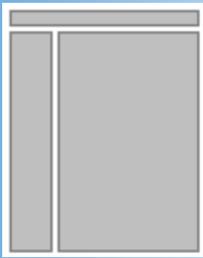
Tabelle 3.6: Vordefinierte target-Werte

Framesets verschachteln

Mit nur einem Frameset kann man ein Browserfenster nur in Zeilen gleicher Spaltenzahl (oder Spalten gleicher Zeilenzahl) aufteilen. Viele interessante Aufteilungen kann man auf diese Weise nicht erzeugen - beispielsweise die Aufteilung in Banner-, Inhaltsverzeichnis- und Hauptframe. Diese werden erst möglich, wenn man Framesets ineinander verschachtelt.

Selbstverständlich sollte man die Verschachtelung nicht übertreiben - schließlich führt die Verschachtelung dazu, dass die einzelnen Frames immer kleiner werden. Bestimmte Verschachtelungen sind aber durchaus typisch und weit verbreitet.

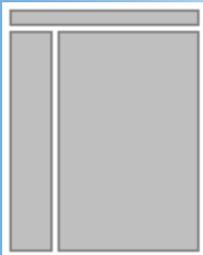
Frame-Aufbau	HTML-Code der Frameseite



In die obere Zeile kommt ein grafisches Banner. Die untere Zeile nimmt den Rest des Browserfensters ein und ist in zwei Spalten geteilt.

Die erste Spalte ist für ein Inhaltsverzeichnis gedacht. Diese Spalte hat eine feste Breite und definiert als Zielframe den neben ihr liegenden Hauptframe, in dem die verschiedenen Seiten des Webs eingeblendet werden.

```
<frameset rows="64,*">
<frame name="Banner" scrolling="no" noresize="noresize" />
<frameset cols="150,*">
<frame name="Inhalt" noresize="noresize"
target="Hauptframe" />
<frame name="Hauptframe" />
</frameset>
</frameset>
```



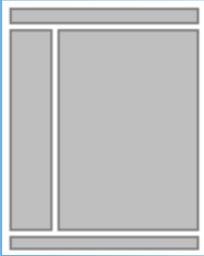
In die obere Zeile kommt eine Navigationsleiste. Je nachdem, welcher Link in der Navigationsleiste gedrückt wird, wechseln die Auswahl der angebotenen Links im Inhaltsverzeichnis (Frame links unten).

Die erste Spalte ist für ein Inhaltsverzeichnis gedacht. Diese Spalte hat eine feste Breite und definiert als Zielframe den neben ihr liegenden Hauptframe, in dem die verschiedenen Seiten des Webs eingeblendet werden.

```
<frameset rows="64,*">
<frame name="Banner" scrolling="no" noresize="noresize"
target="Inhalt" />
<frameset cols="150,*">
<frame name="Inhalt" noresize="noresize"
target="Hauptframe" />
<frame name="Hauptframe" />
</frameset>
</frameset>
```

Bei dieser Seitenaufteilung wäre es schön, wenn bei einem Klick auf einen Link in der Navigationsleiste nicht nur das Inhaltsverzeichnis, sondern auch die

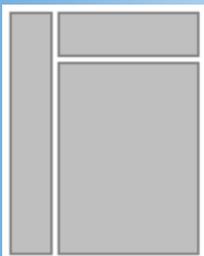
Hauptseite aktualisiert würde. Mit Hilfe von JavaScript ist dies möglich (siehe Kapitel 12.2).



Dieser Aufbau ist eine Erweiterung des obigen Layouts. Hinzugekommen ist eine Fußleiste am unteren Ende, die die gleiche Navigationsleiste wie die Kopfleiste enthält.

```
<frameset rows="64,* ,64">  
  
<frame name="Oben" scrolling="no" noresize="noresize"  
target="Inhalt" />  
  
<frameset cols="150,*">  
  
<frame name="Inhalt" target="Hauptframe" />  
  
<frame name="Hauptframe" />  
  
</frameset>  
  
<frame name="Unten" scrolling="no" noresize="noresize"  
target="Inhalt" />  
  
</frameset>
```

Alternativ könnte man die Fußleiste auch zum Anzeigen von Kontaktinformationen zum Webadministrator nutzen. In diesem Fall würde die Angabe des Zielframes für die Fußleiste entfallen.



Aufteilung in zwei Spalten. In die linke Spalte kommt eine Navigationsleiste (Inhaltsverzeichnis). Je nachdem, welcher Link in der Navigationsleiste gedrückt wird, wechselt die Auswahl der angebotenen Links im Inhaltsverzeichnis (Frame rechts oben). Die rechte Spalte nimmt den Rest des Browserfensters ein und ist in zwei Zeilen geteilt.

Die erste Zeile ist für ein Inhaltsverzeichnis gedacht. Diese Zeile definiert als Zielframe den neben ihr liegenden Hauptframe, in dem die verschiedenen Seiten des Webs eingeblendet werden.

```
<frameset cols="150,*">  
  
<frame name="Links" scrolling="no"  
noresize="noresize"target="ROben" />  
  
<frameset rows="20%,*">  
  
<frame name="ROben" noresize="noresize"target="RUnten" />  
  
<frame name="RUnten" />
```

```
</frameset>
```

```
</frameset>
```

Tabelle 3.7: Browser-Aufteilungen mit verschachtelten Framesets

Probleme mit Browsern ohne Frame-Unterstützung

Es soll sie ja noch geben: älterer Browser, die keine eingebaute Frame-Unterstützung implementiert haben. Es gehört daher zum guten Ton eine alternative Darstellung vorzusehen, die in älteren Browsern statt der Frames angezeigt wird.

Wo definiert man die alternative Anzeige?

Natürlich im HTML-Code der Frameseite, eingeschlossen in das `<noframes>`-Tag. Man kann dies nutzen, um einen Hyperlink zu einer alternativen Seite anzubieten oder sich einfach darauf beschränken, den Websurfer darauf hinzuweisen, was schief gelaufen ist.

```
<frameset cols="20%,80%">
  <frame name="inhalt" src="inhaltsverzeichnis.html" />
  <frame name="hauptframe" src="hauptseite.html" />

<noframes>
  <body>
    <p>Sorry, dies ist eine Frameseite und Ihr Browser unterstützt
      offensichtlich keine Frames</p>
  </body>
</noframes>
</frameset>
```

3.3 Inline-Frames

Frames eröffnen dem Webdesigner eine ganze Reihe von interessanten Gestaltungsmöglichkeiten:

- Frames können wie Tabellen zur Seitenaufteilung genutzt werden.
- Seiteninhalte können partiell aktualisiert werden. Dies vereinfacht beispielsweise den Aufbau von Webs mit Inhaltsverzeichnis und Anzeigebereich. Ohne Frames muss jede Seite, die über die Hyperlinks im Inhaltsverzeichnis geladen werden kann, den kompletten Seitenaufbau mit Inhaltsverzeichnis wiederholen. Mit Frames kann man das Inhaltsverzeichnis in eine eigene HTML-Datei packen und die einzelnen aufgerufenen Seiten beinhalten nur den Text für den Anzeigebereich (hierzu werden wir gleich ein Beispiel sehen).
- Seitenabschnitte, die in Frames eingeschlossen sind, können eigene Bildlaufleisten zugewiesen bekommen. Dies ist beispielsweise interessant, wenn man ein umfangreiches Inhaltsverzeichnis anbietet. Als Frame realisiert, kann man das Inhaltsverzeichnis mit eigenen Bildlaufleisten ausstatten und es vom Anzeigebereich unabhängig scrollen.

Trotz ihrer Vorzüge sind Frames bei Webdesignern nicht unumstritten. Dies hat zum Teil sicherlich historische Gründe (früher gab es keine Möglichkeit, die Rahmenelemente auszublenden), liegt aber auch daran, dass die Platzierung der Frame-Inhalte von den Browsern unterschiedlich gehandhabt wird, so dass es zu Verrückungen kommen kann.

Webs mit aufwendigem grafischem Design verzichten daher oftmals auf den Einsatz von Frames. Leider ging damit bisher auch die Möglichkeit verloren, einzelne Seitenabschnitte mit eigenen Bildlaufleisten zu versehen.

<iframe></iframe>

Die Lösung zu diesem Problem brachte das <iframe>-Tag von HTML 4.

Mit dem <iframe>-Tag kann man einzelne Frames, sogenannte Inline-Frames, in ganz normale Webseiten (ohne Framesets-Definition) einbauen.

Interessant ist dies beispielsweise für Seitenlayouts mit Inhaltsverzeichnis und Anzeigebereich, wenn die über das Inhaltsverzeichnis angesteuerten Dokumente, die in den Anzeigebereich geladen werden, recht lang sind.

Zur Verdeutlichung des Problems lassen Sie uns eine kleine Design-Studie betreiben.



Der Netscape Navigator 4.x unterstützt keine Inline-Frames.

Inhaltsverzeichnisse ohne Frames

In Abschnitt 3.2 haben wir mit Hilfe von Frames ein Web mit Gedichten von Rainer Maria Rilke aufgesetzt. Stellen Sie sich jetzt bitte vor, dieses Web soll ein aufwendiges grafisches Design erhalten, dass sich mit Frames nur unbefriedigend lösen lässt. Wir entscheiden uns daher für ein Tabellen-Design.

Die Seiten unseres Webs bestehen aus einem Kopfbereich, einem Inhaltsverzeichnis und einem Anzeigebereich. Kopfbereich und Inhaltsverzeichnis sind auf allen Seiten gleich, der Anzeigebereich variiert - je nachdem, welcher Link im Inhaltsverzeichnis angeklickt wurde.



Abbildung 3.10: Grund-Design, das allen Seiten gemeinsam ist (das Tabellengitter ist nur zur Verdeutlichung der Tabellenstruktur eingeblendet)

1. Wir beginnen mit der Hauptseite *tabelle.html*. Wir stellen sie soweit fertig, dass sie alle Elemente enthält, die den Seiten des Webs gemeinsam sind: das Tabellenlayout, den Kopfbereich und das Inhaltsverzeichnis (siehe Abbildung 3.10). (Aus didaktischen Gründen besteht der Kopfbereich aus einer normalen Überschrift und das Inhaltsverzeichnis ist eine Definitionsliste. Das »aufwendige, grafische Design«, wegen dem wir auf die Verwendung von Frames verzichten, denken Sie sich bitte dazu. ;-) .)

Den Anzeigebereich, der für jede Seite verschieden ist, sparen wir erst einmal aus.

Die Hyperlinks weisen auf die noch anzulegenden Seiten: *panther.html*, *herbsttag.html* und *karussell.html*.

Listing 3.6: tabelle.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <title>Rilke-Gedichte</title>
</head>
<body>
<table border="0" width="700" cellpadding="10" cellspacing="0">
<colgroup span="2">
    <col width="200">
    <col width="500">
</colgroup>
<tr>
```

```

<td colspan="2">
  <h1>Ausgewählte Rilke-Gedichte</h1>
</td>
</tr>
<tr>
  <td width="200" valign="top">
    <dl>
      <dt>Rilke</dt>
      <dd><a href="panther.html">Der Panther</a></dd>
      <dd><a href="herbsttag.html">Herbsttag</a></dd>
      <dd><a href="saal.html">Im Saal</a></dd>
      <dd><a href="karussell.html">Das Karussell</a></dd>
    </dl>
  </td>
  <td width="500" valign="top">
    &nbsp;
  </td>
</tr>
</table>
</body>
</html>

```



Die Breite der Spalten für das Inhaltsverzeichnis und den Anzeigebereich geben wir auf zwei verschiedene Weisen an: durch die colgroup-Definition und durch die width-Angabe in den <td>-Tags der Spalten. Letzter Weg wird zwar vom HTML-Standard als nicht empfehlenswert (deprecated) eingestuft, ist aber derzeit immer noch der einzig verlässliche Weg sicherzustellen, dass alle wichtigen Browser die Breitenangaben wie gewünscht berücksichtigen.

2. Als Nächstes erstellen wir die Webseiten, die über die Hyperlinks geladen werden. Da wir nicht mit Frames arbeiten, wird die neu geladene Webseite das gesamte Browserfenster füllen und die aktuelle Datei komplett ersetzen. Die Webseite muss also die allen Webseiten gemeinsamen HTML-Elemente plus den individuellen HTML-Code für den Anzeigebereich enthalten. Beginnen wir mit der Seite *panther.html*.

Wir legen eine Kopie von *tabelle.html* an und speichern diese unter dem Namen *panther.html* ab. Damit übernehmen wir in einem Arbeitsgang das vorgegebene Layout, Design und die Navigationsstruktur.

Wenn die Seite *panther.html* aufgerufen wird, soll das Gedicht »Der Panther« in den Anzeigebereich eingeblendet werden. Wir fügen also den HTML-Code zur Anzeige des Gedichts in die Tabellenzelle für den Anzeigebereich ein:

```

<table border="0" width="700" cellpadding="10" cellspacing="0">
...

  <td width="500" valign="top">
    <h2>DER PANTHER</h2>
    <pre>
Sein Blick ist vom Vorübergehn der Stäbe
so müd geworden, daß er nichts mehr hält.
Ihm ist, als ob es tausend Stäbe gäbe
und hinter tausend Stäben keine Welt.
Der weiche Gang geschmeidig starker Schritte,
der sich im allerkleinsten Kreise dreht,
ist wie ein Tanz von Kraft um eine Mitte,
in der betäubt ein großer Wille steht.
Nur manchmal schiebt der Vorhang der Pupille
sich lautlos auf - dann geht ein Bild hinein,
geht durch der Glieder angespannte Stille -
und hört im Innern auf zu sein.
    </pre>
  </td>
</tr>
</table>

```

Auf die gleiche Weise legen wir die Webseiten für die restlichen Gedichte an (die Texte finden Sie in den gleichnamigen HTML-Seiten auf der Buch-CD).

3. Zum Schluss kann man noch den Anzeigebereich der Startseite *tabelle.html* mit einem Begrüßungstext oder Ähnlichem füllen.

Fertig! Laden Sie die Seite *tabelle.html* jetzt in Ihren Browser und folgen Sie den Hyperlinks. Lesen Sie auch das Gedicht »Das Karussell«. Dieses Gedicht ist etwas länger. Wenn man es ganz liest, muss man die Anzeige des Browserfensters soweit herunterscrollen, dass die Hyperlinks aus dem Inhaltsverzeichnis danach verschwunden sind. Um also das nächste Gedicht auswählen zu können, muss man wieder an den Beginn der Webseite hochscrollen - was nicht sonderlich benutzerfreundlich ist. Um dieses Manko zu beheben, gibt es verschiedene Möglichkeiten:

- man kann unter der letzten Zeile des Gedichts zwei Hyperlinks zum nächsten und zum vorangehenden Gedicht einfügen,
- man kann an das Ende der Webseite eine zusätzliche Navigationsleiste einfügen (in Form einer dritten Tabellenzeile).
- man kann den Anzeigebereich mit Hilfe eines Inline-Frames realisieren.

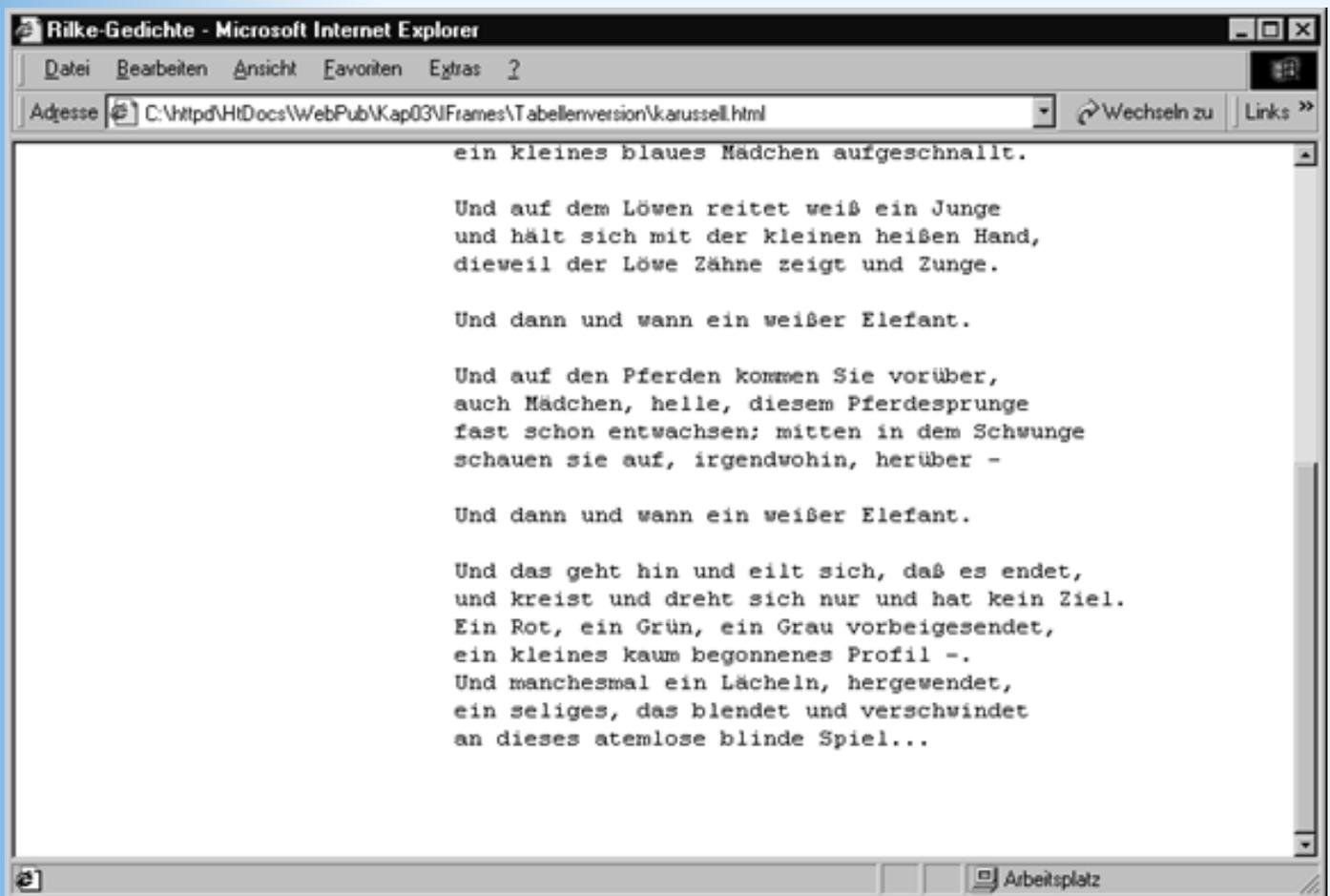


Abbildung 3.11: Ooops, wo geht es jetzt weiter?



Das oben dargestellte Design hat gegenüber der Verwendung von Frames auch noch einen weiteren Nachteil. Werden Änderungen an den gemeinsamen Seitenteilen erforderlich (soll beispielsweise ein weiteres Gedicht in das Inhaltsverzeichnis aufgenommen werden), müssen alle Seiten des Webs aktualisiert werden.

Inhaltsverzeichnisse mit Inline-Frames

Inline-Frames (<iframe>) werden genauso wie normale Frames (<frame>) definiert - nur dass sie nicht Teil einer Frameset-Deklaration sein müssen, sondern an beliebiger Stelle im <body>-Bereich einer normalen Webseite definiert werden.

Was bedeutet dies für unsere Sammlung von Rilke-Gedichten?

Wir können das »aufwendige grafische Design« wie gewünscht mit Hilfe einer Tabelle realisieren und trotzdem von den Vorteilen der Frames profitieren (eigene Bildlaufleiste, über Hyperlinks ansteuerbar), indem wir in die Tabellenzelle für den Anzeigebereich einen Inline-Frame einfügen.



Inline-Frames werden derzeit leider noch nicht vom Netscape Navigator unterstützt.

1. Zuerst setzen wir wieder die Hauptseite *tabelle.html* mit dem Tabellenlayout, dem Kopfbereich und dem Inhaltsverzeichnis auf (siehe Abbildung 3.10). Wir können dazu die Vorlage aus dem vorangehenden Abschnitt verwenden (legen Sie ein neues Verzeichnis an und kopieren Sie einfach die Datei *tabelle.html*).

In den Anzeigebereich fügen wir einen Inline-Frame ein:

```
<iframe name="anzeige" src="blank.html" width="450" height="350"
    scrolling="auto" frameborder="0">
    <p>Sorry, Ihr Browser unterstützt keine Inline-Frames</p>
</iframe>
```

Im Header-Abschnitt legen wir den Inline-Frame *anzeige* als Standardziel für die Frames der Seite fest.

```
<head>
    <title>Rilke-Gedichte</title>
    <base target="anzeige">
</head>
```

Der vollständige HTML-Code der Startseite sieht damit wie folgt aus:

Listing 3.7: Die Startseite *tabelle.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <title>Rilke-Gedichte</title>
    <base target="anzeige">
</head>
<body>
<table border="0" width="700" cellpadding="10" cellspacing="0">
<colgroup span="2">
    <col width="200">
    <col width="500">
</colgroup>
<tr>
    <td colspan="2">
        <h1>Ausgewählte Rilke-Gedichte<h2>
    </td>
```

```

</tr>
<tr>
  <td width="200" valign="top">
    <dl>
      <dt>Rilke</dt>
      <dd><a href="panther.html" >Der Panther</a></dd>
      <dd><a href="herbsttag.html">Herbsttag</a></dd>
      <dd><a href="saal.html">Im Saal</a></dd>
      <dd><a href="karussell.html">Das Karussell</a></dd>
    </dl>
  </td>
  <td width="500" valign="top">
    <iframe name="anzeige" src="blank.html" width="450" height="350"
      scrolling="auto" frameborder="0">
      <p>Sorry, Ihr Browser unterstützt keine Inline-Frames</p>
    </iframe>
  </td>
</tr>
</table>
</body>
</html>

```

2. Als Nächstes erstellen wir die Webseiten, die über die Hyperlinks geladen werden. Da wir die Seiten in einen Frame einblenden, brauchen die Webseiten wieder nur den HTML-Code für den Anzeigebereich (sprich die Gedichte) zu enthalten. Beginnen wir mit der Seite *panther.html*:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Rilke-Gedichte</title>
</head>
<body>
<h2>DER PANTHER</h2>
<pre>
  Sein Blick ist vom Vorübergehn der Stäbe
  so müd geworden, daß er nichts mehr hält.
  Ihm ist, als ob es tausend Stäbe gäbe
  und hinter tausend Stäben keine Welt.
  Der weiche Gang geschmeidig starker Schritte,
  der sich im allerkleinsten Kreise dreht,
  ist wie ein Tanz von Kraft um eine Mitte,
  in der betäubt ein großer Wille steht.
  Nur manchmal schiebt der Vorhang der Pupille
  sich lautlos auf - dann geht ein Bild hinein,
  geht durch der Glieder angespannte Stille -
  und hört im Innern auf zu sein.
</pre>

```

```
</body>  
</html>
```

Auf die gleiche Weise legen wir die Webseiten für die restlichen Gedichte an (die Texte finden Sie in den gleichnamigen HTML-Seiten auf der Buch-CD).

3. Zum Schluss setzen wir noch eine leere Webseite *blank.html* auf. Bei der Definition des Inline-Frames haben wir diese Seite als Wert für das `src`-Attribut angegeben. Der Browser wird diese Datei also in den Frame laden, wenn die Startseite *tabelle.html* zum ersten Mal aufgerufen wird.

Listing 3.8: Eine leere Webseite - blank.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"  
  "http://www.w3.org/TR/html4/strict.dtd">  
<html>  
<head>  
  <title>Rilke-Gedichte</title>  
</head>  
<body>  
</body>  
</html>
```



Statt einer leeren Webseite könnte man anfänglich natürlich auch einen Begrüßungstext oder Ähnliches laden.



Abbildung 3.12: Mit Inline-Frames kann man einzelne Seitenabschnitte mit Bildlaufleisten ausstatten

3.4 Steuerelemente und Formulare

Wer im Internet schon einmal etwas bestellt oder sich für das Download kostenloser Demoversionen bei einem Software-Anbieter hat registrieren lassen, dem dürfte die Verwendung von Formularen auf HTML-Seiten bereits wohl vertraut sein. Wo immer Daten von Webbesuchern an den Server (und damit an den Webbetreiber) übermittelt werden sollen, stößt man auf Formulare: Bestellformulare, Registrierformulare, Formulare zur Eintragung in Gästebücher.

Als Websurfer füllen Sie die Formulare Schritt für Schritt aus: Sie tippen Ihre Daten in die dafür vorgesehenen Eingabefelder, Sie aktivieren oder deaktivieren Optionsfelder, Sie wählen aus den Vorgaben eines Listenfeldes, und zum Schluss klicken Sie auf den Abschicken-Schalter, um Ihre Eingaben an den Webserver zu schicken.

Schwieriger als das Ausfüllen und Abschicken eines Formulars ist es ein Formular aufzusetzen. Formulare bestehen immer aus zwei Teilen: der Eingabemaske, die dem Webbesucher angezeigt wird, und dem Programmcode, der die Formulareingaben verarbeitet, wenn der Webbesucher den Abschicken-Schalter drückt. In diesem Kapitel werden wir uns erst einmal auf das Aufsetzen der Eingabemaske beschränken. In den Kapiteln 11 und 17 können wir uns dann ganz auf die Bearbeitung der Formulareingaben konzentrieren.

Die HTML-Steuererelemente, die wir aus den Formularen kennen, können aber auch außerhalb von Formularen verwendet werden - beispielsweise zum Aufbau von dynamischen Benutzerschnittstellen. Es lohnt sich also, zuerst einmal die HTML- Steuererelemente für sich zu betrachten.



Der Netscape Navigator zeigt Steuererelemente nur dann an, wenn Sie in <form>-Tags gefasst sind. Wir fassen daher in diesem Buch grundsätzlich alle Steuererelemente in <form></form>-Tags ein - auch wenn wir gar kein richtiges Formular (mit Abschicken-Schalter) erstellen wollen und laut HTML-Standard eigentlich auf die <form>-Tags verzichten könnten.

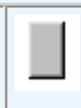
HTML-Steuererelemente

Der Standardweg zur Erzeugung von Steuererelementen führt über das <input>-Tag. Mit Ausnahme von Auswahllisten (Menüs) und Labeln können alle HTML-Steuererelemente mit dem <input>-Tag erzeugt werden.

Steuererelemente erzeugen

Das <input>-Tag kennt eine ganze Reihe von Attributen. Zwei dieser Attribute sind besonders wichtig und sollten für alle Steuererelemente definiert werden: name und type.

```
<input type="button" name="schalter1" />
```



- Über das type-Attribut gibt man an, welche Art von Steuererelement angezeigt werden soll. Mögliche Werte für dieses Attribut sind: text, password, checkbox, radio, submit, reset, file, hidden, image, button, wobei text der Standardwert ist, der verwendet wird, wenn Sie keinen eigenen Typ angeben.
- Den Namen können wir frei vergeben. Er sollte - innerhalb des Formulars (oder der Webseite, wenn das Steuererelement außerhalb von <form>-Tags definiert ist) - eindeutig sein. Über den Namen können Skripte und CGI-Programme auf das Steuererelement zugreifen.



Der Netscape Navigator zeigt Steuererelemente nur dann an, wenn Sie in <form>-Tags gefasst sind.

Steuerelemente konfigurieren

Für die meisten Steuerelementtypen kennt das `<input>`-Tag zusätzliche Attribute, über die man das Steuerelement konfigurieren kann. Eines dieser Attribute ist das `value`-Attribut, über das man dem Steuerelement einen Wert zuweisen kann. Speziell für Schalter gibt man über dieses Attribut den Schaltertitel an.

```
<input type="button" name="schalter1"
value="Klick mich!" />
```



Wenn Sie für einen Schalter das `value`-Attribut setzen, passt der Browser die Größe des Schalters an seinen Titel an.

Es gibt noch einige andere Attribute, die man zur Konfiguration der Steuerelemente verwenden kann. Allerdings werden diese Attribute nicht immer von allen Browsern gleichermaßen gut unterstützt (zu dem Zeitpunkt, da dieses Buch geschrieben wurde, bot der Internet Explorer die beste Unterstützung).

Attribut	Beschreibung
<code>alt= "Vornamen eingeben"</code>	Kurze Beschreibung für Browser, die keine Grafiken und Steuerelemente anzeigen können.
<code>tabindex="1"</code>	Zur Festlegung der Reihenfolge, in der Steuerelemente über die (Tab)-Taste angesteuert werden können. Steuerelemente mit niedrigeren, positiven Werten werden zuerst angesteuert, ansonsten (kein Wert vorgegeben oder mehrere Steuerelemente mit gleichen Werten) gilt die Definitionsreihenfolge.
<code>disabled="disabled"</code>	Das Steuerelement wird deaktiviert (graue Darstellung, kann nicht verwendet werden)
<code>accesskey="V"</code>	Zur Festlegung eines Tastaturkürzels (einzelnes Zeichen) für ein Steuerelement. Tippt der Besucher der Webseite das entsprechende Zeichen ein, erhält das Steuerelement, das dieses Zeichen als Tastaturkürzel verwendet, den Eingabefokus.

Tabelle 3.8: Allgemeine Steuerelement-Attribute

Welche weiteren Attribute für welche Steuerelemente verwendet werden können, entnehmen Sie bitte Tabelle 3.9 im nachfolgenden Abschnitt.

Ereignisbehandlung für Steuerelemente

Steuerelemente in HTML-Seiten sind ziemlich sinnlos, wenn man nicht auch irgendwie darauf reagiert, wenn der Anwender mit dem Steuerelement interagiert. Eigentlich ist dies das Thema der Kapitel 11 und 17, doch sollten wir uns zum Austesten der Steuerelemente durchaus den

Spaß gönnen, die Steuerelemente mit ein wenig dynamischem HTML zu verbinden. Dazu bedarf es zwei kleiner Änderungen:

Im Header-Abschnitt müssen wir die verwendete Skriptsprache angeben:

```
<meta http-equiv="Content-Script-Type" content="text/javascript">
```

Im `<input>`-Tag verbinden wir eines der Steuerelementereignisse mit JavaScript-Code, der ausgeführt wird, wenn das Steuerelement ausgewählt wird. Zur Demonstration wählen wir das `onclick`-Ereignis (wird ausgelöst, wenn das Steuerelement angeklickt wird) und verbinden es mit JavaScript-Code, der eine Erfolgsmeldung ausgibt:

```
<input type="button" name="schalter1"
      value="Klick mich!" onclick="alert('Treffer')" />
```

Der vollständige Quelltext einer Testseite könnte jetzt wie folgt aussehen:

Listing 3.9: schalter.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Schalter</title>
  <meta http-equiv="Content-Script-Type" content="text/javascript">
</head>
<body>
<h2>Schalter als Beispiel für ein Steuerelement</h2>
<form>
<input type="button" name="schalter1"
      value="Klick mich!" onclick="alert('Treffer')" />
</form>
</body>
</html>
```



Abbildung 3.13: Beim Klicken auf den Schalter wird eine Meldung angezeigt

Die verschiedenen Steuerelemente

Die meisten Steuerelemente können wie gesagt mit Hilfe des `<input>`-Tags erzeugt werden, für andere (mehrzeiliges Textfeld, Label) gibt es eigene Tags. Wie die verschiedenen Steuerelemente erzeugt werden und welche spezifischen Attribute für die verschiedenen Steuerelemente gesetzt werden können, zeigt die folgende Tabelle.

Formularfeld	Beschreibung
Einzeiliges Textfeld	<pre><input type="text" name="T1" value="hier eintippen" size="20" maxlength="40" /></pre> <p>Steuerelement, in das der Webbesucher einen Text eingeben kann (beispielsweise seinen Namen).</p> <p>Mit <code>value</code> kann ein Text im Textfeld angezeigt werden.</p> <p>Mit <code>size</code> kann man die Größe des Elements in Textzeichen angeben, <code>maxlength</code> ist die maximale Anzahl von Zeichen, die man in das Textfeld eingeben kann. Ist <code>maxlength</code> größer als <code>size</code> scrollt der Browser gegebenenfalls die Anzeige.</p> <p>Mit <code>readonly="readonly"</code> kann man verhindern, dass der Leser den Text im Textfeld verändern kann.</p>

Passwortfeld	<pre><input type="password" ... /></pre> <p>Variante des einzeiligen Textfeldes, bei dem die Eingabe jedoch verschlüsselt wird (statt der eingegebenen Zeichen sieht man auf dem Bildschirm nur Sternchen)</p>
Mehrzeiliges Textfeld	<pre><textarea name="S1" rows="2" cols="20"></textarea></pre> <p>Mehrzeiliges Eingabefeld - beispielsweise für Kommentare, Diskussionsbeiträge, etc. Über die Attribute rows und cols kann man angeben, wie viele Zeilen und Spalten (Zeichen in einer Zeile) gleichzeitig (ohne Scrollen) sichtbar sein sollen.</p> <p>Mit readonly="readonly" kann man verhindern, dass der Leser den Text im Textfeld verändern kann.</p>
Schaltfläche	<pre><input type="button" name="B1" value="Schaltfläche" /></pre> <p>Schaltfläche, die beim Anklicken irgendwelche Aktionen ausführt. Über value gibt man den Titel des Schalters an.</p> <p>Alternativ kann man Schalter auch mit dem <button>-Element erzeugen:</p> <pre><button type="button" value="1" name="B3"></pre> <p>Klick mich!</p> <pre></button></pre> <p>In diesem Fall wird der Titel des Schalters zwischen den <button>-Tags angegeben (und kann auch ein Bild enthalten). Über value gibt man einen Wert an, der an den Server geschickt wird, wenn der Schalter gedrückt wird (und ein Programm zur Verarbeitung der Formulardaten vorgesehen ist).</p>
Abschicken-Schalter	<pre><input type="submit" name="B2" value="Abschicken" /></pre> <p>Schaltfläche zum Abschicken von Formulardaten (siehe Kapitel 11 und 17).</p>

Zurücksetzen-Schalter	<pre><input type="reset" name="B3" value="Zurücksetzen" /></pre> <p>Schaltfläche zum Zurücksetzen von Formulardaten (siehe Kapitel 11 und 17).</p>
Kontrollkästchen	<pre><input type="checkbox" name="C1" value="1" checked="checked" /></pre> <p>Zum Aktivieren und Deaktivieren einzelner Optionen.</p> <p>Durch Setzen des Booleschen Attributs checked kann man vorgeben, dass das Steuerelement anfangs gesetzt ist.</p> <p>Über value kann man das Steuerelement mit einem Wert verbinden.</p>
Optionsfeld	<pre><input type="radio" value="1" checked="checked" name="R1" /></pre> <p>Zum Aktivieren und Deaktivieren einzelner Optionen. Optionsfelder können im Gegensatz zu Kontrollkästchen intern gruppiert werden, wobei innerhalb einer Gruppe immer nur eine Option ausgewählt werden kann. Optionsfelder werden gruppiert, indem man allen Optionsfeldern einer Gruppe den gleichen Namen zuweist (in obigem Beispiel also R1).</p> <p>Durch Setzen des Booleschen Attributs checked kann man vorgeben, dass das Steuerelement anfangs gesetzt ist.</p> <p>Über value kann man das Steuerelement mit einem Wert verbinden.</p>
Auswahlfeld	<pre><select multiple="multiple" size="1" name="D1"> <option>Bananen</option> <option selected="selected">Äpfel</option> <option>Orangen</option> </select></pre> <p>Aufklappbare Liste, die eine Reihe verschiedener Optionen zur Auswahl bereitstellt.</p> <p>Wenn das Boolesche Attribut multiple gesetzt ist, können mehrere Optionen gleichzeitig ausgewählt werden. Das Attribut size gibt an, wie</p>

	<p>viele Optionen in der nicht aufgeklappten Liste angezeigt werden.</p> <p>Über das Boolesche Attribut selected kann man festlegen, welche Optionen anfangs ausgewählt sein sollen.</p> <p>Versieht man die Optionen mit label-Attributen und ordnet sie in <optgroup>-Elementen an, kann man hierarchische Auswahlfelder erzeugen (siehe HTML 4-Standard). Die gängigen Browser unterstützen derzeit noch keine hierarchischen Auswahlfelder.</p>
Bild	<pre><input type="image" name="I1" src="bild.gif" /></pre> <p>Zur Anzeige von Bildern und Grafiken in Formularen.</p> <p>Zur Unterstützung von ImageMaps stehen die Attribute usemap und ismap zur Verfügung.</p> <p>Da dieses Steuerelement jedoch nicht von allen Browsern unterstützt wird, empfiehlt es sich, Bilder auch in Formularen mit dem -Tag einzubinden.</p>
Datei	<pre><input type="file" name="T1" /></pre> <p>Zur Auswahl einer Datei, die an den Server geschickt werden soll (siehe auch Erläuterungen zum enctype-Attribut in Abschnitt 3.4.2).</p>
Beschriftung	<pre><label>Beschriftung</label></pre> <p>Zur Beschriftung anderer Steuerelemente.</p> <p>Wenn das andere Steuerelement über eine id-Kennung verfügt, kann man das Label über sein for-Attribut mit dem anderen Steuerelement verbinden.</p> <pre><label for="vorname">Vorname : </label></pre> <pre><input type="text" name="T1" id="vorname" size="20"></pre> <p>Wenn ein solches Label den Fokus erhält, gibt es diesen an sein zugeordnetes Steuerelement weiter.</p>

Tabelle 3.9: Typen von Steuerelementen

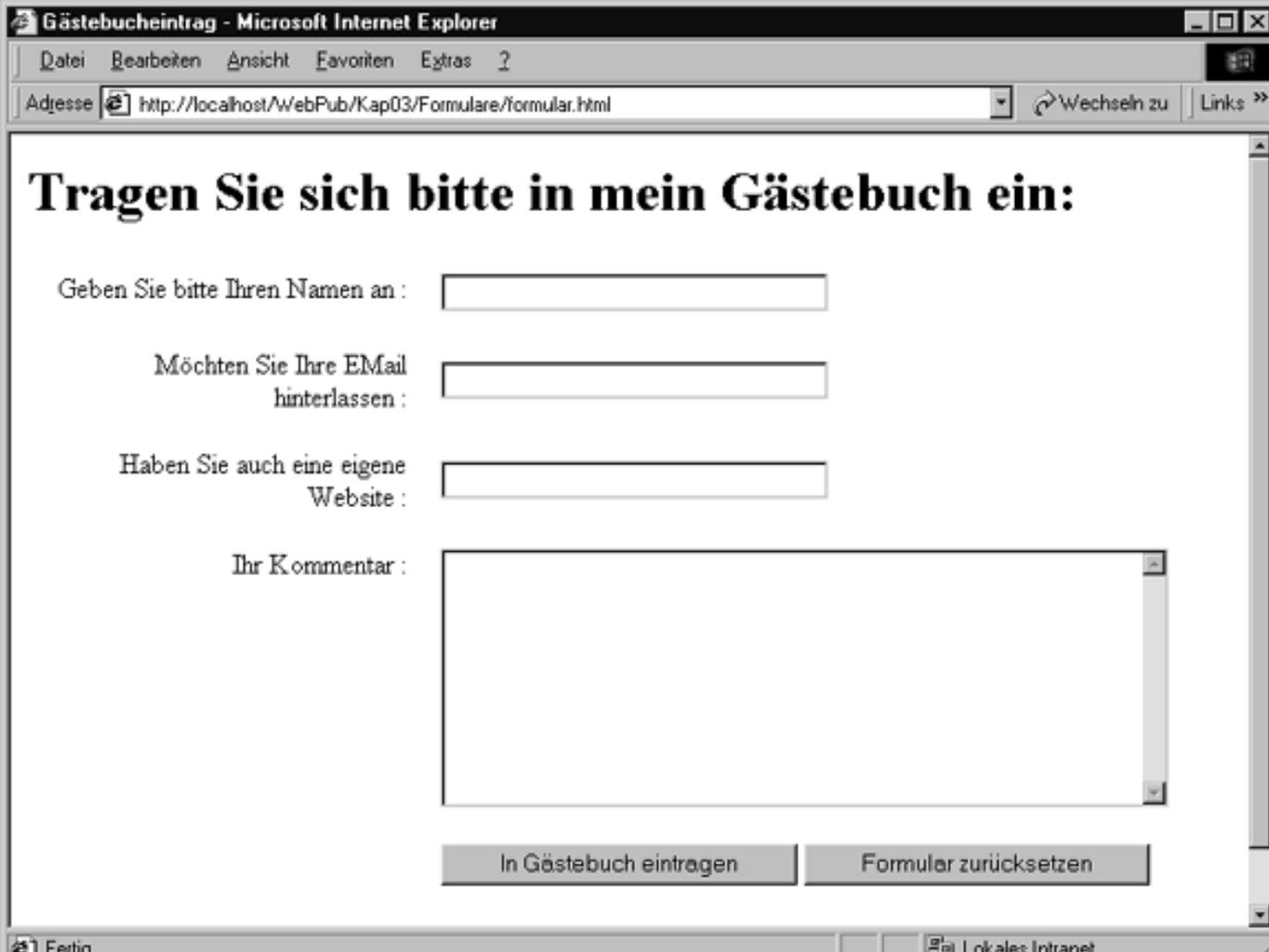


Wenn Sie sich anschauen wollen, wie die einzelnen Steuerelemente in Ihrem

Browser aussehen, laden Sie einfach die Datei `steuerelemente.html` von der Buch-CD

HTML-Formulare

Wenn man die HTML-Befehle zur Erzeugung von Steuerelementen kennt, stellt der Aufbau von Formularen keine Schwierigkeit mehr dar. Betrachten Sie einmal das Formular aus Abbildung 3.14, über das sich Besucher einer Website in ein Gästebuch eintragen können.



The screenshot shows a Microsoft Internet Explorer window titled "Gästebucheintrag - Microsoft Internet Explorer". The address bar displays "http://localhost/WebPub/Kap03/Formulare/formular.html". The main content area features a heading "Tragen Sie sich bitte in mein Gästebuch ein:". Below the heading are four input fields: a text box for "Geben Sie bitte Ihren Namen an:", a text box for "Möchten Sie Ihre EMail hinterlassen:", a text box for "Haben Sie auch eine eigene Website:", and a larger text area for "Ihr Kommentar:". At the bottom of the form are two buttons: "In Gästebuch eintragen" and "Formular zurücksetzen". The status bar at the bottom indicates "Fertig" and "Lokales Intranet".

Abbildung 3.14: Formular zur Eintragung in ein Gästebuch

Schauen wir uns an, wie man dieses Formular in drei bis vier Schritten erstellen kann.

Steuerelemente aufsetzen

Zuerst richten wir die Steuerelemente für die gewünschten Benutzereingaben ein:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Gästebucheintrag</title>
</head>
```

```

<body>
<h1>Tragen Sie sich bitte in mein Gästebuch ein:</h1>
<!-- Hier beginnen die Steuerelemente -->
<p>Geben Sie bitte Ihren Namen an      :
    <input name="name" size="30" maxlength="50" /></p>
<p>Möchten Sie Ihre EMail hinterlassen :
    <input name="email" size="30" maxlength="50" /> </p>
<p>Haben Sie auch eine eigene Website  :
    <input name="website" size="30" maxlength="50" /></p>
<p>Ihr Kommentar :</p>
<p><textarea name="kommentar" rows="9" cols="50"></textarea></p>
</body>
</html>

```

Umwandlung der Steuerelemente in ein Formular

Bisher haben wir nur eine Sammlung von Steuerelementen, die von verschiedenen Browsern noch nicht einmal angezeigt würde. Um aus diesen Steuerelementen ein Formular zu machen, fassen wir Sie in `<form>`-Tags ein:

```

<form action="/cgi-bin/programm.pl" method="get">
...
</form>

```

Um aus einer Sammlung von Steuerelementen ein Formular zu machen (dass auch vom Netscape Navigator angezeigt wird und das man mit JavaScript-Skripten auswerten kann), braucht man im `<form>`-Starttag keine Attribute anzugeben. Die meisten Formulare (Bestellformulare, Registrierformulare, Eingabeformulare für Gästebücher) sind aber darauf angewiesen, dass die vom Besucher der Website eingetippten Daten über das Internet zurück an den Server geschickt werden.

Will man die Formulareingaben über das Internet zurück an den Server schicken, muss man mindestens zwei weitere Angaben machen:

- Wer soll die Formulareingaben entgegen nehmen und auswerten. Üblicherweise ist der Empfänger ein CGI-Programm (siehe Kapitel 17), dessen URL als Wert des `action`-Attributs angegeben wird.
- Auf welche Weise soll der Browser die Formulardaten weiterreichen. Hierfür gibt es zwei Möglichkeiten: Nach der GET-Methode hängt der Browser die Formulardaten in codierter Form an den URL des Empfängerprogramms an und schickt den URL zusammen mit den angehängten Daten an den Server. Nach der POST-Methode baut der Browser die codierten Formulardaten in den Rumpf der HTTP-Anfrage ein, die er an den Server schickt. Die technischen Details brauchen uns im Moment allerdings nicht zu interessieren. Wichtig ist, dass Sie dem Attribut `method` als Wert entweder "get" oder "post" zuweisen müssen. Welchen Wert Sie angeben, hängt zum Teil davon ab, welche Methode das Empfängerprogramm unterstützt (auch dazu mehr in Kapitel 17).

```

<form action="/cgi-bin/gaestebuch.pl" method="get">

```

...
</form>



Der Vollständigkeit halber sei erwähnt, dass man auch noch die Codierung der Formulareingaben durch den Browser einstellen kann. Der HTML-Standard sieht dafür das Attribut `enctype` vor, dem man zwei Werte zuweisen kann: `"application/x-www-form-urlencoded"` (der Standardwert) und `"multipart/form-data"` (für Dateiuploads, siehe `file-Steuer`element).

Schalter zum Abschicken des Formulars hinzufügen

Wie kann der Besucher unserer Webseite das Formular abschicken? Natürlich indem er den Schalter zum Abschicken anklickt. Dazu müssen wir diesen Schalter aber erst einmal einrichten.

Springen Sie zum Ende des Formulars und richten Sie einen Schalter zum Abschicken - und eventuell auch einen Schalter zum Zurücksetzen - des Formulars ein. Wie in Tabelle 3.9 aus Abschnitt 3.4.1 erwähnt, handelt es sich dabei um ganz normale Schalter, deren `type`-Attribut auf `submit` beziehungsweise `reset` gesetzt wird.

```
<form action="/cgi-bin/gaestebuch.pl" method="get">
<p>Geben Sie bitte Ihren Namen an      :
  <input name="name" size="30" maxlength="50" /></p>
<p>Möchten Sie Ihre EMail hinterlassen :
  <input name="email" size="30" maxlength="50" /> </p>
<p>Haben Sie auch eine eigene Website  :
  <input name="website" size="30" maxlength="50" /></p>
<p>Ihr Kommentar :</p>
<p><textarea name="kommentar" rows="9" cols="50"></textarea></p>
<input type="submit" value="In Gästebuch eintragen">
<input type="reset" value="Formular zurücksetzen">
</form>
```

Formatierung

An sich ist unser Formular jetzt fertig, doch es sieht im Browser nicht sonderlich gut aus (siehe Abbildung 3.15).

Tragen Sie sich bitte in mein Gästebuch ein:

Geben Sie bitte Ihren Namen an :

Möchten Sie Ihre EMail hinterlassen :

Haben Sie auch eine eigene Website :

Ihr Kommentar :

In Gästebuch eintragen

Formular zurücksetzen

Abbildung 3.15: Unformatiertes Formular

Der schnellste Weg die Beschriftungen und Steuerelemente eines Formulars schön und übersichtlich aneinander auszurichten, führt meist über eine Tabelle. Der folgende HTML-Code erzeugt beispielsweise das Formular aus Abbildung 3.14.

Listing 3.10: Formular aus formulare.html

```
<h1>Tragen Sie sich bitte in mein Gästebuch ein:</h1>
<form action="/cgi-bin/gaestebuch.pl" method="get">
  <table border="0" cellspacing="0" cellpadding="10">
    <colgroup span=2">
      <col width="230">
      <col width="450">
    </colgroup>
    <tr>
      <td align="right" valign="top" width="230">
        Geben Sie bitte Ihren Namen an      : </td>
      <td><input name="name" size="30" maxlength="50" /></td>
    </tr>
    <tr>
      <td align="right" valign="top" width="230">
        Möchten Sie Ihre EMail hinterlassen : </td>
      <td><input name="email" size="30" maxlength="50" /></td>
    </tr>
    <tr>
      <td align="right" valign="top" width="230">
```

```

        Haben Sie auch eine eigene Website : </td>
    <td><input name="website" size="30" maxlength="50" /></td>
</tr>
<tr>
    <td align="right" valign="top" width="230">Ihr Kommentar :</td>
    <td><textarea name="kommentar" rows="9" cols="50"></textarea></td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td><input type="submit" value="In Gästebuch eintragen" />
        <input type="reset" value="Formular zurücksetzen" /></td>
</tr>
</table>
</form>

```

3.5 Zusammenfassung

Tabellen werden in HTML mit Hilfe des `<table>`-Tags definiert. Ein typisches Tabellengerüst hat folgenden grundlegenden Aufbau:

```

<table border="1" width="550" >
    <caption>Dies ist die Überschrift</caption>
    <tr>
        <th>Überschrift Spalte 1.</th>
        <th>Überschrift Spalte 2</th>
    </tr>
    <tr>
        <td>Feld 11</td>
        <td>Feld 12</td>
    </tr>
    <tr>
        <td>Feld 21</td>
        <td>Feld 22</td>
    </tr>
    <tr>
        <td>Feld 31</td>
        <td>Feld 32</td>
    </tr>
</table>

```

Tabellen können in vielfältiger Weise konstruiert und formatiert werden. So kann man beispielsweise auch den Rahmen und das Gitternetz der Tabelle ausblenden (`<table border="0">`, je nach Bedarf auch noch `cellspacing` und `cellpadding` auf 0 setzen) - ein Umstand der dazu geführt hat, dass Tabellen zu einem beliebten und wichtigen Mittel zur Gestaltung des Seitenlayouts wurden.

Ein weiteres, ebenfalls sehr interessantes Mittel zur Gestaltung des Seitenlayouts ist die Aufteilung in Frames. Hierbei wird der `<body>`-Abschnitt der Webseite durch die in `<frameset>`-

Tags gefasste Definition der Frames ersetzt.

```
<frameset cols="20%,80%">
  <frame name="inhalt" src="inhaltsverzeichnis.html">
  <frame name="hauptframe" src="hauptseite.html">

  <noframes>
    <p>Sorry, dies ist eine Frameseite und Ihr Browser unterstützt
      offensichtlich keine Frames</p>
  </noframes>
</frameset
```

Über das target-Argument des <a>-Tags kann man steuern, in welchen Frame die Webseite, auf die der Link verweist, geladen werden soll. Definiert man das target- Argument im <base>-Tag einer Webseite, legt man den angegebenen Frame als Standard- Zielframe für die Hyperlinks der Webseite fest.

Zum Abschluss dieses Tages haben wir uns noch den Aufbau von Formularen und die Verwendung von Steuerelementen angesehen.

3.6 Fragen und Antworten

Frage:

Der Browser verändert immer meine Vorgaben für die Breite der Tabellenspalten. Wie kann ich das verhindern?

Antwort:

In Übereinstimmung mit dem HTML-Standard hat der Browser das Recht die Breitenangaben an den Inhalt der Zellen anzupassen. Die verschiedenen Browser machen davon in unterschiedlichem Maße Gebrauch. Wenn Sie dieses Verhalten gänzlich unterbinden wollen, müssen Sie die Stileigenschaft table-layout auf fixed setzen.

```
<table width="700" cellspacing="10px" style="table-layout: fixed">
```

Antwort:

Allerdings wird diese Stileigenschaft derzeit nur vom Netscape 6-Browser unterstützt.

Frage:

Wie kann ich den Text einer Zelle von den Rändern der Zelle abrücken?

Antwort:

Wenn Sie global für die gesamte Tabelle festlegen wollen, dass der Inhalt der Zellen von allen Rändern der Zelle gleichermaßen abzurücken ist, verwenden Sie das <table>-Attribut cellpadding:

```
<table border="1" width="100%" cellpadding="10">
```

Internet Explorer 5 und Netscape 6 unterstützen zudem die Stileigenschaften

padding und margin, mit denen Sie den Inhalt einzelner Zellen von deren Rändern abrücken können. Folgende Anweisung rückt den Zelleinhalt beispielsweise 20 Pixel vom linken Rand und 10 Pixel vom oberen Rand ab (mehr zu den Eigenschaften padding und margin in Kapitel 4.3.2):

```
<td width="450" style="padding-left: 20px; padding-top: 10px">
```

Frage:
Kann man Tabellen ineinander verschachteln?

Antwort:
Ja, selbstverständlich.

Frage:
Ich verwende Frames ohne Rahmen (border="0", frameborder="0"). Trotzdem reicht der Inhalt meiner Frames nicht bis an den Rand des Browserfenster heran. Mache ich etwas falsch?

Antwort:
Nein. Sie müssen nur im <body>-Tag der Webseite, die in den Frame geladen werden soll marginwidth="0" marginheight="0" topmargin="0" und leftmargin="0" setzen (sonst erzeugen manche Browser-Versionen einen Standardrand). Hinweis: Man kann marginheight=0 marginwidth=0 auch in <frame> einfügen, aber Netscape Navigator 4 gibt nichtsdestotrotz noch einen Pixel Abstand dazu.

3.7 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

Quiz

1. Mit welchem Tag definiert man eine Tabellenzeile, mit welchem Tag eine Tabellenzelle?
2. Wie kann man zwei benachbarte Zellen zu einer Zelle verschmelzen?
3. Kann man in einer Zelle einer bestehenden Tabelle eine neue Tabelle anlegen?
4. Wie zentriert man eine Tabelle auf der Webseite?
5. Wie zentriert man den Inhalt einer Tabellenzelle?
6. Wie teilt man Webseiten in Frames auf?
7. Wie kann man die Rahmen der Frames ausblenden?
8. Wie kann man verhindern, dass der Websurfer die Rahmen verschiebt?
9. Welche Möglichkeiten gibt es, den Zielframe eines Hyperlinks festzulegen?

Übungen

1. Welcher HTML-Code könnte folgende Tabelle erzeugen?.

Abbildung 3.16: Stellen Sie diese Tabelle nach.

2. Wie erzeugt man folgenden Seitenaufbau?



Abbildung 3.17: Nachzustellendes Layout

3. Überarbeiten Sie den HTML-Code der Tabelle aus *bananenproduktion.html* (Listing 3.1) so, dass die Tabelle im Internet Explorer und Netscape Navigator möglichst gleich dargestellt wird.
4. Vervollständigen Sie das Beispiel aus Abschnitt 3.2.4, indem Sie das Inhaltsverzeichnis mit entsprechenden Hyperlinks versehen.
5. Sie wollen auf Ihrer Webseite eine Umfrage durchführen. Da die Internet-Autobahnen immer mehr verstopfen, plädieren Sie für den Internet-freien Sonntag. In einem passenden Umfrageformular können die Besucher Ihrer Website abstimmen, ob sie diesen Vorschlag gutheißen oder ablehnen.

Tag 4

Formatieren mit Stylesheets

Stylesheets sind keine ganz neue Erfindung. Tatsächlich gibt es sie schon seit einigen Jahren, nur dass sie früher - wegen mangelnder Unterstützung durch die Webbrowser - von den Webautoren kaum beachtet wurden. Was nutzt einem Webautor aber ein Tool, mit dem er sich zwar ganz neue Möglichkeiten zur Formatierung und Gestaltung seiner Webseiten erschließt, dieser Fortschritt aber von den Besuchern seiner Website gar nicht wahrgenommen wird?

Nun, zum Glück hat sich in dieser Hinsicht manches geändert. Seit der 4er-Version unterstützen der Internet Explorer wie auch der Netscape Navigator die Formatierung und Positionierung mit Stylesheets - wenn auch zum Teil in recht unterschiedlichem Maße. So ist die Unterstützung durch den Netscape Navigator etwas lückenhaft und weicht in manchen Details vom CSS-Standard ab, während der Internet Explorer den CSS-Standard in fast schon vorbildlicher Weise implementiert hat (auch wenn er eine Reihe von proprietären Elementen eingefügt hat). Nun, man hat bei Netscape Besserung versprochen und der neue Navigator 6, der jetzt gerade auf den Markt kommt, unterstützt die offiziellen Webstandards wesentlich besser. Dem verspäteten Triumphzug der Stylesheets sollte also nichts mehr im Wege stehen.

Die Themen heute:

- Was sind Stylesheets und welche Vorteile bringen Sie uns?
- Wie verbindet man Stylesheets mit HTML-Dokumenten?
- Wie setzt man Stylesheet-Definitionen auf?
- Die wichtigsten Stileigenschaften
- Das Box-Modell von HTML und CSS
- Größenangaben in Stylesheets
- Pseudoelemente für Hyperlinks
- Wie funktionieren Vererbung und Cascading?
- Wie wird man mit der unterschiedlichen Unterstützung durch die Browser fertig?

4.1 Wozu braucht man Stylesheets?

Wie Sie wissen, verwenden Webdokumente die sogenannten HTML-Tags und deren Attribute zur Formatierung und zur Festlegung des Seitenlayouts. Die meisten dieser Tags sind allerdings nur mehr oder weniger grobe Vorgaben, deren genauen Umsetzung im Ermessen des Webbrowsers liegt. Zudem ist die Zahl der HTML-Tags recht beschränkt. Zwar kamen in den letzten Jahren immer neue Tags und Attribute hinzu, doch sind der Formatierung mit Tags nach wie vor erhebliche Grenzen gesetzt.

Hier setzt die Idee der Stylesheets an. Stylesheets räumen dem Webdesigner mehr Möglichkeiten zur dezidierten Gestaltung seiner Webseiten ein und vereinfachen und organisieren gleichzeitig die Erstellung und Wartung der Seiten. Dabei stellen Stylesheets nichts grundsätzlich Neues dar; im Grunde funktionieren Sie ähnlich wie HTML- Attribute.

Warum Sie Stylesheets nutzen sollten:

- Mit Cascading Stylesheets können Sie das Erscheinungsbild ganzer Webseiten frei und flexibel gestalten.
- Mit Cascading Stylesheets können Sie einzelne Elemente von Webseiten präzise positionieren und ganz nach ihren Wünschen formatieren.
- Cascading Stylesheets können modular aufgebaut werden, das heißt, Sie können neue Stylesheets auf der Grundlage bereits definierter Stylesheets definieren.
- Cascading Stylesheets können wie selbstdefinierte Formatvorlagen verwendet werden. Wenn Sie also häufiger Textstellen eine bestimmte Kombination von Formatbefehlen zuweisen müssen, lohnt es sich, diese Befehle in einem Stylesheet zu definieren, um dann nur noch in einem Schritt das Stylesheet zuweisen zu müssen.
- Cascading Stylesheets können die Zahl der für eine Website verwendeten Formatbefehle reduzieren - dies kann dem Autor Arbeitszeit und dem Site-Besucher Download-Zeit sparen.

Bevor wir uns ein wenig eingehender mit den Stylesheets, mit der dahinter stehenden CSS-Spezifikation und den Möglichkeiten zum sinnvollen Einsatz von Stylesheets befassen, möchte ich Ihnen anhand eines einfachen einführenden Beispiels ein Gefühl dafür geben, was Stylesheets wirklich sind, wie sie eingesetzt werden und welche Vorteile sie uns bringen.

Vorbereitung des Beispiels



Abbildung 4.1: Demoseite zur Formatierung mit Stylesheets

Legen Sie eine neue HTML-Seite an mit vier gleichen Textzeilen, die Sie mit Hilfe des `<h2>`-Tags als Überschriften der zweiten Ebene formatieren (siehe Abbildung 4.1).

Listing 4.1: ueberschriften1.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <title>Formatieren mit Stylesheets</title>
</head>
<body>

<h2>Dies ist eine Überschrift der zweiten Ebene</h2>
<h2>Dies ist eine Überschrift der zweiten Ebene</h2>
```

```
<h2>Dies ist eine Überschrift der zweiten Ebene</h2>
<h2>Dies ist eine Überschrift der zweiten Ebene</h2>
</body>
</html>
```

Textfarbe per HTML-Attribut

Wie kann man einer Überschrift eine andere Textfarbe zuweisen? Versuchen wir es zuerst mit den normalen HTML-Tags. Wir wollen uns dabei einmal ganz bewusst auf die Ebene eines blutigen HTML-Anfängers stellen, damit Sie einen umso deutlicheren Eindruck von den Vorteilen der Formatierung mit Stylesheets bekommen (und der etwas konfuse, traditionellen Formatierung mit HTML-Attributen).

Wenn Sie Kapitel 2.8.1 aufmerksam gelesen haben und über ein gutes Gedächtnis verfügen, werden Sie sich erinnern, dass es für das `<body>`-Tag ein Attribut `text` gibt, mit dessen Hilfe man die Standardtextfarbe für den gesamten Inhalt einer Webseite festlegen kann:

```
<body text="#00FFFF">
...
</body>
```

In unserem Beispiel wollen wir aber nur die Textfarbe der ersten `<h2>`-Überschrift ändern. Eine naheliegende Möglichkeit wäre, das Attribut `text` für die `<h2>`-Tags zu verwenden:

```
<h2 text="#00FFFF">Dies ist eine Überschrift der zweiten Ebene</h2>
```

Bei der Begutachtung im Browser stellt man dann allerdings schnell fest, dass es so nicht geht. Der Grund hierfür ist, dass das `text`-Attribut nur für das `<body>`-Tag und nicht für die Überschriften-Tags definiert ist.

Also laden wir uns die aktuelle HTML-Spezifikation herunter und suchen nach einem geeigneten Formatierungsattribut. Schließlich finden wir das gewünschte Attribut: es heißt `color` und ist... leider auch nicht für das `<h2>`-Tag definiert. Tatsächlich kann es nur zusammen mit dem ``-Tag verwendet werden (und `<basefont>`). Also verfallen wir auf einen Trick: Wir schließen den Text der ersten Überschrift zusätzlich in ``-Tags ein und legen in dem öffnenden ``-Tag die Textfarbe fest. (Wir geben die Textfarbe im Beispiel als RGB-Wert an, siehe Kapitel 2.8.2)

```
<h2><font color="#00FFFF">Dies ist eine Überschrift der zweiten
Ebene</font></h2>
```



Bei dieser Gelegenheit sei darauf hingewiesen, dass man HTML-Tags gemäß XHTML nicht überlappen darf (also nicht `<h2>Dies ... </h2>` schreiben).



Abbildung 4.2: Unterschiedlich formatierte Überschriften

Textfarbe per Stildefinition

Gleiches kann man auch mit dem style-Attribut erreichen. Das style-Attribut kann auf alle HTML-Tags im <body>-Bereich angewendet werden (einschließlich dem <body>-Tag). Die Textfarbe des betreffenden HTML-Elements wird über die Stileigenschaft color gesetzt:

```
<h2 style="color: #00FFFF">Dies ist eine Überschrift der zweiten Ebene</h2>
```

Was ist daran so fortschrittlich?

HTML kennt zwar eine Reihe von Attributen, mit denen man HTML-Elemente formatieren kann, doch ist nicht jedes HTML-Attribut für jedes HTML-Element verfügbar. Beispielsweise können Sie das Attribut color, wie oben geschehen, dem -Tag zuweisen, Sie können es aber nicht direkt dem <h2>-Tag zuweisen!

Innerhalb einer Stildefinition können Sie dagegen alle für Stylesheets verfügbaren Stileigenschaften verwenden. (Welche Eigenschaften dies im Einzelnen sind, werden wir uns in Abschnitt 4.3 anschauen.) Und über das Attribut style können Sie jedes Stylesheet jedem beliebigen HTML-Element zuweisen. Im Gegensatz zu den inkonsistenten und lückenhaften Formatierungsmöglichkeiten der HTML-Tags und -Attribute bieten Stylesheets daher ein einfaches, leistungsfähiges und wohl konstruiertes Formatierungsmodell. Kein Wunder also, dass die meisten HTML-Tags und Attribute zur reinen Textformatierung seit dem HTML 4.0-Standard als »deprecated« (abzulehnen) eingestuft sind und zugunsten von Stylesheets aufgegeben werden sollten.

Stylesheets können aber noch mehr.

Eine Stildefinition für alle Überschriften der Ebene 2

Wenn Sie alle Überschriften der Ebene 2 in einer speziellen Farbe, beispielsweise in Rot, anzeigen wollen, kann die Zuweisung der Farbe an jede einzelne Überschrift recht aufwendig werden (speziell wenn Sie viele Überschriften der Ebene 2 verwendet haben).

Stylesheets können da schnell Abhilfe schaffen.

Statt im <body>-Bereich nach den <h2>-Tag zu suchen und diese um ein entsprechendes style-Attribut zu erweitern, setzen wir im <head>-Abschnitt des HTML-Dokuments eine Stylesheet-Definition auf, in der wir

festlegen, dass alle Überschriften der Ebene 2 als Textfarbe Rot verwenden sollen.

```
<head>
  <title>Formatieren mit Stylesheets</title>
  <style>
    h2 { color: red }
  </style>
</head>
```

Mit Hilfe der Tags `<style>...</style>` betten wir das Stylesheet in den Header-Abschnitt des Webdokuments ein. Dann geben wir das HTML-Tag an, dessen Formatierung wir beeinflussen wollen (im Beispiel h2) und hängen in geschweiften Klammern eine durch Semikolons getrennte Liste von Stileigenschaften an (im Beispiel nur `color: red`).

Der vollständige Quelltext unserer Testseite sieht dann wie folgt aus.

Listing 4.2: ueberschriften4.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Formatieren mit Stylesheets</title>
  <style>
    h2 { color: red }
  </style>
</head>
<body>

<h2 style="color: #00FFFF">Dies ist eine Überschrift der zweiten Ebene</h2>
<h2>Dies ist eine Überschrift der zweiten Ebene</h2>
<h2>Dies ist eine Überschrift der zweiten Ebene</h2>
<h2>Dies ist eine Überschrift der zweiten Ebene</h2>
</body>
</html>
```



Abbildung 4.3: ueberschriften4. html

In Abbildung 4.3 sind die Farben leider verloren gegangen, aber in Ihrem Browser dürften Sie nun zweierlei feststellen:

- Die unteren drei Überschriften erscheinen in rotem Text.
- Die erste Überschrift ist nach wie vor zyanblau.

Wir können daraus schließen, dass die Stildefinition im Header-Teil der Webseite standardmäßig auf alle Überschriften der Ebene 2 (<h2>-Tag) angewendet wird. Wird im Body-Teil für eine Überschrift der Ebene 2 explizit eine andere Textfarbe vorgesehen, hat diese Einstellung Vorrang vor der Stildefinition für die ganze Seite.

4.2 Stylesheets definieren und zuweisen I

Stile - im Sinne der CSS-Spezifikation - bestehen aus einer Folge von Stileigenschaften und deren Werte, wobei die einzelnen Stileigenschaften durch Semikolons getrennt werden:

```
eigenschaft1: wert; eigenschaft2: wert; ...
```

Eine solche Folge von Stileigenschaften kann auf verschiedene Weisen definiert werden:

- Inline-Stile
- Eingebettete Stylesheets
- Externe Stylesheets (die zu bevorzugende Weise, mit der wir uns allerdings erst in Abschnitt 4.4 beschäftigen).

Inline-Stile

Im einfachsten Fall weist man die Stildefinition mit Hilfe des style-Attributs einem einzelnen HTML-Element im <body>-Bereich zu:

```
<p style="color: red; text-align: right">
```

Diese Art der Formatierung haben wir ja bereits in Kapitel 2 kennen gelernt, hier noch einmal ein kleines Beispiel:

Listing 4.3: inline.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Formatieren mit Stylesheets</title>
</head>
<body>
```

```
<p style="color: red; font-family: sans-serif; text-align: right">Dieser Absatz
verwendet eine Schrift ohne Serifen (idealerweise Arial), erscheint in roter
Textfarbe und ist rechtsbündig ausgerichtet</p>
```

```
<p>Dieser Absatz erscheint in Standardschrift. Lediglich <span style="text-
decoration: underline">diese Textpassage</span> ist durch Unterstreichung
hervorgehoben.
```

```
<div style="margin-left: 10em; margin-right: 10em; margin-top: 4em; text-align:
```

```
justify">
<h2>Blocksatz</h2>
<p>Dieser Abschnitt (von der Überschrift bis zum nächsten Absatz) wird vom
vorangehenden Absatz durch einen Leerraum von 4 em abgetrennt, links und rechts
eingezogen und an beiden Seiten bündig dargestellt. </p>
<p>Um dies zu erreichen, wurden die Überschrift und die beiden Absätze in ein
DIV-Element eingeschlossen.</p>
</div>
</body>
</html>
```



Abbildung 4.4: Formatierung mit Inline-Stilen

Eingebettete Stylesheets

Wie Sie aus Abschnitt 4.1 bereits wissen, kann man Stylesheets auch im <head>-Abschnitt des HTML-Dokuments definieren. Diese Technik bringt erhebliche Vorteile mit sich:

- die Formatierungsanweisungen stehen übersichtlich am Anfang des HTML- Dokuments
- man kann mit einer Stildefinition alle Vorkommen eines bestimmten HTML- Elements formatieren (siehe einführendes Beispiel)
- zusammen genommen bedeuten die beiden vorangehenden Punkte, dass man die Formatierung einer Webseite, die eingebettete Stylesheets verwendet, ohne großen Aufwand ändern und überarbeiten kann

Wir werden die Bedeutung dieser Punkte im Laufe dieses Kapitels noch etwas deutlicher herausarbeiten. Zuerst aber wollen wir uns anschauen, wie man Stylesheets in HTML- Dokumente einbettet.

Stylesheets in HTML-Dokumente einbetten

Zum Einbetten von Stylesheets in HTML-Dokumente stellt uns HTML das Tag <style> zur Verfügung.

Beachten Sie, dass Stylesheet-Definitionen mit `<style>` nur im `<head>`- Abschnitt des HTML-Dokuments vorkommen dürfen:

```
<head>
  <title>Formatieren mit Stylesheets</title>
  <style>
    ...
  </style>
</head>
```

Zu dem `<style>`-Tag gibt es zwei Attribute: `type` und `media`.

Es wurde bereits darauf hingewiesen, dass es derzeit zwar nur eine wirklich bedeutende Stylesheet-Spezifikation gibt (CSS), dass der HTML-Standard aber ausdrücklich auch andere Stylesheet-Spezifikationen zulässt. Da die Syntax und die zur Verfügung stehenden Stileigenschaften von der Stylesheet-Spezifikation abhängen, müssen wir dem Browser mitteilen, welcher Stylesheet-Spezifikation unsere Stylesheets folgen. Hierzu kann man das `type`-Attribut verwenden.

```
<style type="text/css">
```

Diese Deklaration teilt dem Browser beispielsweise mit, dass unsere Stylesheets der CSS- Spezifikation folgen.

Natürlich macht die Angabe der Stylesheet-Spezifikation nur Sinn, wenn der Browser die betreffende Stylesheet-Spezifikation kennt und unterstützt. Da dies derzeit nur für CSS- Stylesheets der Fall ist und HTML als Standard-Stylesheetsprache sowieso `text/css` vorgibt, ist die `type`-Angabe im Grunde unnötig. Sie gehört aber zum guten Stil und wird in Zukunft vielleicht noch an Bedeutung gewinnen.

Anstatt die Standard-Stylesheetsprache über das `type`-Attribut in den Stylesheet- Definitionen anzugeben, kann man Sie auch vorab mit einem entsprechenden META-Tag festlegen:

```
<meta http-equiv="content-style-type" content="text/css">
```

Diese Vorgehensweise hat zudem den Vorteil, dass sie auch für Inline-Stile gültig ist.



Wenn Sie Inline-Stile verwenden, sollten Sie stets obiges META-Tag definieren.

Medientypen und Stylesheets

HTML-Dokumente können nicht nur von Webbrowsern auf Computer-Bildschirmen dargestellt werden. Prinzipiell können HTML-Dokumente von beliebigen Ausgabegeräten verarbeitet werden, solange es nur ein entsprechendes Programm gibt, das wie ein Webbrowser den HTML-Code einliest und entsprechend dem verwendeten Ausgabegerät aufbereitet. In diesem Sinne definiert der HTML-Standard derzeit bereits folgende Medientypen, die Sie als Werte für das `media`-Attribut angeben können.

screen (normaler Grafik-Bildschirm), tty (Terminal), tv (TV-Bildschirm), projection (Projektor), handheld (kleine Bildschirme wie bei PalmPilots), print (Drucker), braille (für Blinde), aural (mit Sprachengenerator) und all (Alle).

Durch die Angabe des Medientyps können Sie festlegen, dass das betreffende Stylesheet nur für die Ausgabe auf einem entsprechenden Mediengerät gedacht ist. Die CSS- Stylesheet-Spezifikation kennt übrigens für verschiedene dieser Medientypen eigene Stileigenschaften. Wenn Sie mehr darüber erfahren möchten, schlagen Sie bitte in der CSS2-Spezifikation nach.

Zurück zu unserer Stylesheet-Definition.

Zwischen den <style>-Tags können wir nun beliebig viele Stildefinitionen einfügen. Jede solche Stildefinition besteht aus einer Liste von durch Semikolons getrennten Stileigenschaften (ganz wie bei den Inline-Stilen) und wird in geschweifte Klammern eingefasst. (Die geschweiften Klammern helfen dem Browser zu erkennen, wo eine Stildefinition beginnt und wo sie endet.)

```
<head>
...
<style type="text/css">
    {color: red; font-family: sans-serif; text-align: right}
    {text-decoration: underline}
</style>
</head>
```

Jetzt gibt es nur noch ein Problem. Wie legt man fest, welche Stildefinition zur Formatierung welcher Elemente im HTML-Dokument verwendet werden sollen? Im Falle der Inline-Stile war dies klar, da der Inline-Stil sich natürlich auf das HTML-Element bezieht, für das er definiert wird. Bei der Stylesheet-Definition im <head>-Abschnitt ist dies nicht so klar. Wir müssen also irgendwie eine Beziehung zwischen unseren Stildefinitionen und den zu formatierenden HTML-Elementen im <body>-Abschnitt herstellen. Dies geschieht mit Hilfe sogenannter Selektoren, die den Stildefinitionen vorangestellt werden.

Es gibt drei Typen von Selektoren

- HTML-Tag-Selektoren
- class-Selektoren und
- ID-Selektoren

Die Selektoren werden den Stildefinitionen einfach vorangestellt. Auf diese Weise erhalten die Stildefinitionen so etwas wie einen Namen, über den man die Stildefinitionen unten im <body>-Abschnitt auswählen kann. Mehr dazu gleich im nächsten Abschnitt.

```
<head>
...
<style type="text/css">
    p {color: red; font-family: sans-serif; text-align: right}
    .unter {text-decoration: underline}
</style>
</head>
```

Zum Schluss noch zwei Anmerkungen zur Browserunterstützung.

Ältere und neuere Browser

Ältere Browser (beispielsweise Internet Explorer und Netscape Navigator vor Version 4) verstehen keine Stylesheet-Definitionen. Um sicherzustellen, dass diese Browser die Stylesheet-Definitionen nicht als Text darstellen (manche Browser stellen unbekannte Inhalte im Header-Bereich tatsächlich als Inhalt der Webseite dar), kann man CSS- Stylesheetdefinitionen auskommentieren:

```

<head>
...
<style type="text/css">
<!--
  p {color: red; font-family: sans-serif; text-align: right}
  .unter {text-decoration: underline}
-->
</style>
</head>

```

Browser, die CSS-Stylesheets unterstützen, werden die Stildefinitionen trotz der Kommentare erkennen und berücksichtigen.

Obwohl diese Form der Auskommentierung gängige Praxis ist und von allen aktuellen Browsern verstanden wird (und wohl auch noch für längere Zeit verstanden wird), ergibt sich hier ein Problem bei der Umstellung von HTML zu XHTML (siehe Kapitel 21). XHTML-Dokumente werden nämlich wie alle XML-Dokumente von einem XML-Parser analysiert (der in diesem Fall in die Webbrowser integriert ist), und es gibt derzeit keine verlässliche Zusicherung, dass der XML-Parser den Code in den Kommentaren nicht einfach löscht. Des Weiteren wird der XML-Parser Zeichen wie < (<) als HTML- Sonderzeichen und nicht als normale Zeichen interpretieren. Man kann diesem Problem begegnen, indem man die Deklaration des eingebetteten Stylesheets in <![CDATA[und]]> einfasst:

```

<head>
<style type="text/css">
<![CDATA[
  p {color: red; font-family: sans-serif; text-align: right}
  .unter {text-decoration: underline}
]]>
</style>
</head>

```

Noch besser ist es allerdings, externe Stylesheets zu verwenden (siehe Abschnitt 4.4).

Selektoren

Selektoren dienen dazu, eine Verbindung zwischen den Stildefinitionen eines eingebetteten Stylesheets und den zu formatierenden HTML-Elementen herzustellen. Die einfachste Form der Verbindung ist dabei der HTML-Selektor.

HTML-Selektoren



HTML-Selektoren lauten wie die entsprechenden HTML-Tags ohne die spitzen Klammern.

Stildefinitionen, die innerhalb eines Stylesheets an ein HTML-Tag als Selektor zugewiesen werden, gelten für alle HTML-Elemente auf der Webseite, die das betreffende HTML-Tag verwenden. Auf diese Weise kann man beispielsweise alle Überschriften einer Ebene mit einem Schlag einheitlich formatieren:

```
h1 { color: rgb(255,0,255) }
```

Oder man rückt alle Absätze links und rechts ein:

```
p { margin-left: 10%; margin-right: 10%}
```

Listing 4.4: eingebettet1.html - Stylesheets mit HTML-Selektoren

```
<head>
  <title>Stylesheets</title>
  <style type="text/css">
    h1 { color: rgb(255,0,255); font-family: Arial;
        font-size: 20pt }
    h2 { font-size: 16pt }
    p { margin-left: 10%; margin-right: 10%}
  </style>
</head>
```

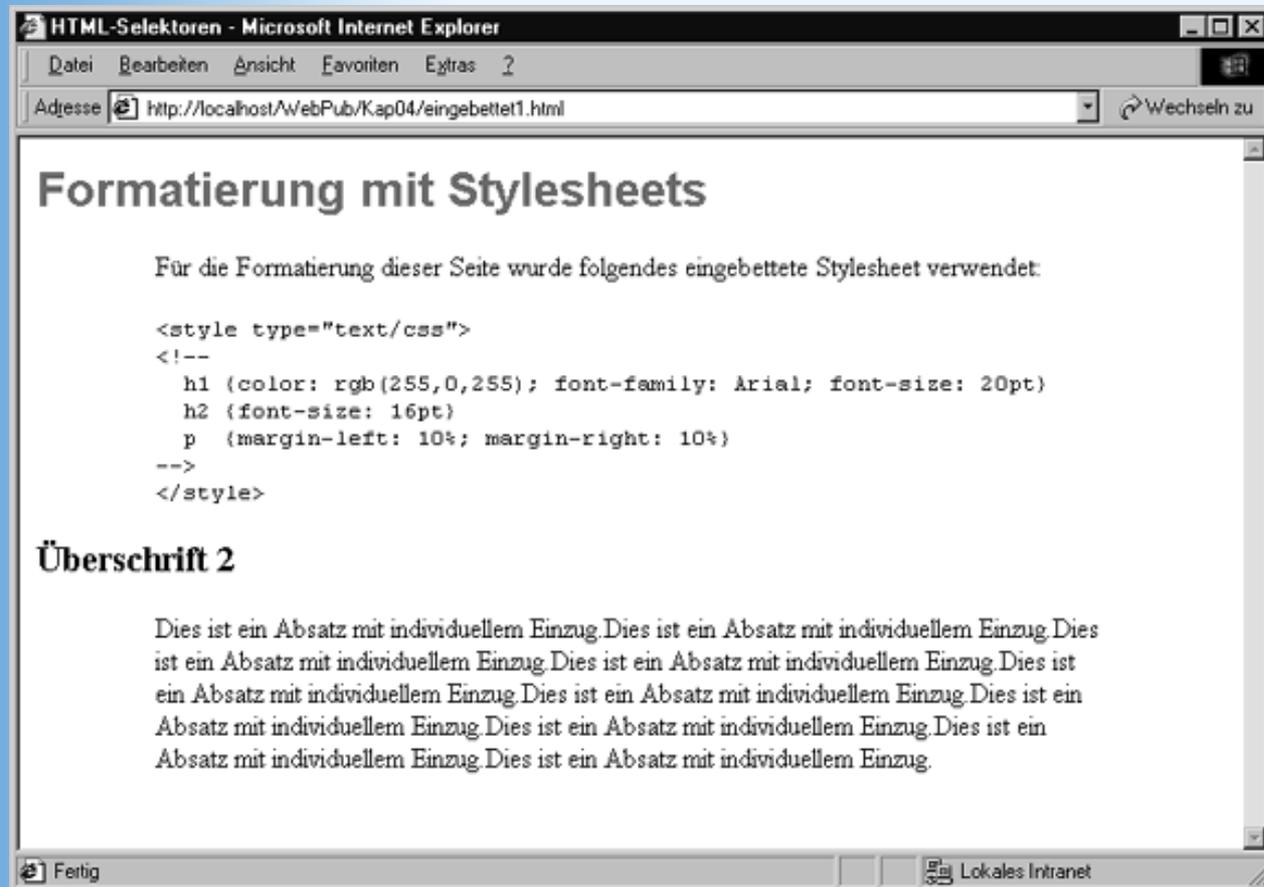


Abbildung 4.5: HTML-Selektoren

Wenn Sie eine Stildefinition mit mehreren HTML-Tags verbinden wollen, brauchen Sie die Stildefinition nicht mehrfach aufzusetzen. Geben Sie einfach die betreffenden HTML- Tags als kommasetrennte Liste an:

```
h1, h2, h3 {color: rgb(255,0,255); font-family: Arial; font-size: 20pt }
```

Sie können auf diese Weise beispielsweise eine allgemeine Grundformatierung für verwandte HTML-Tags vorgeben und diese durch nachfolgende Tag-spezifische Deklarationen anpassen. So legt der folgende Code fest, dass alle Überschriften der Ebenen 1, 2 und 3 in Rosa und der Schrift Arial darzustellen sind.

Die Schriftgröße ist dagegen für alle Ebenen unterschiedlich:

```
h1, h2, h3 {color: rgb(255,0,255); font-family: Arial; font-size: 24pt}
h2          {font-size: 20pt}
h3          {font-size: 16pt}
```

class-Selektoren

Die Formatierung über HTML-Selektoren ist recht einfach und bequem, hat aber unter Umständen den Nachteil, dass der zugewiesene Stil ausnahmslos auf alle Vorkommen des betreffenden HTML-Tags angewendet wird. Soll eine Stildefinition dagegen nur gezielt für zwei oder drei spezielle Vorkommen des Tags gelten (beispielsweise einzelne <p>- Absätze) oder soll einem Teil der <p>-Absätze ein anderer Stil zugewiesen als den restlichen <p>-Absätzen oder möchte man eine Stildefinition auf einen speziellen <p>-Absatz, ein -Tag und eine weiter unten folgende Liste anwenden, wird es schwierig. Hier kommt uns das Konzept der class-Selektoren zu Hilfe.



Ein class-Selektor ist ein beliebiger Name zur Bezeichnung einer Stildefinition, dem ein Punkt vorangeht.

```
.stilname {color: rgb(255,0,255); font-family: Arial}
```

Nachdem der Stil definiert und mit einem class-Namen verbunden ist, kann man den Stil mit Hilfe des class-Attributs den HTML-Elementen zuweisen, die gemäß diesem Stil formatiert werden sollen:

```
<body>
...
<p class="stilname">
...
<ol class="stilname">
...
</body>
```

So entsteht eine Gruppe (Klasse) von einzelnen HTML-Elementen, die alle mit dem gleichen Stil formatiert werden.

In Abbildung 4.5 ist es zum Beispiel etwas störend, dass alle Absätze (HTML-Tag <p>) gemäß der Stildefinition `p { margin-left: 10%; margin-right: 10%; }` links und rechts eingezogen werden. Dem kann abgeholfen werden, indem man die Einzüge an einen class-Selektor (`.einzug`) bindet

```
.einzug { margin-left: 10%; margin-right: 10%; }
```

und diesen class-Selektor dann mit Hilfe des Attributs `class` nur den Absätzen zuweist, die eingezogen werden sollen (im Beispiel dem Absatz unter der Überschrift 2):

```
<p class="einzug">
```

Das Ergebnis sehen Sie in Abbildung 4.6.

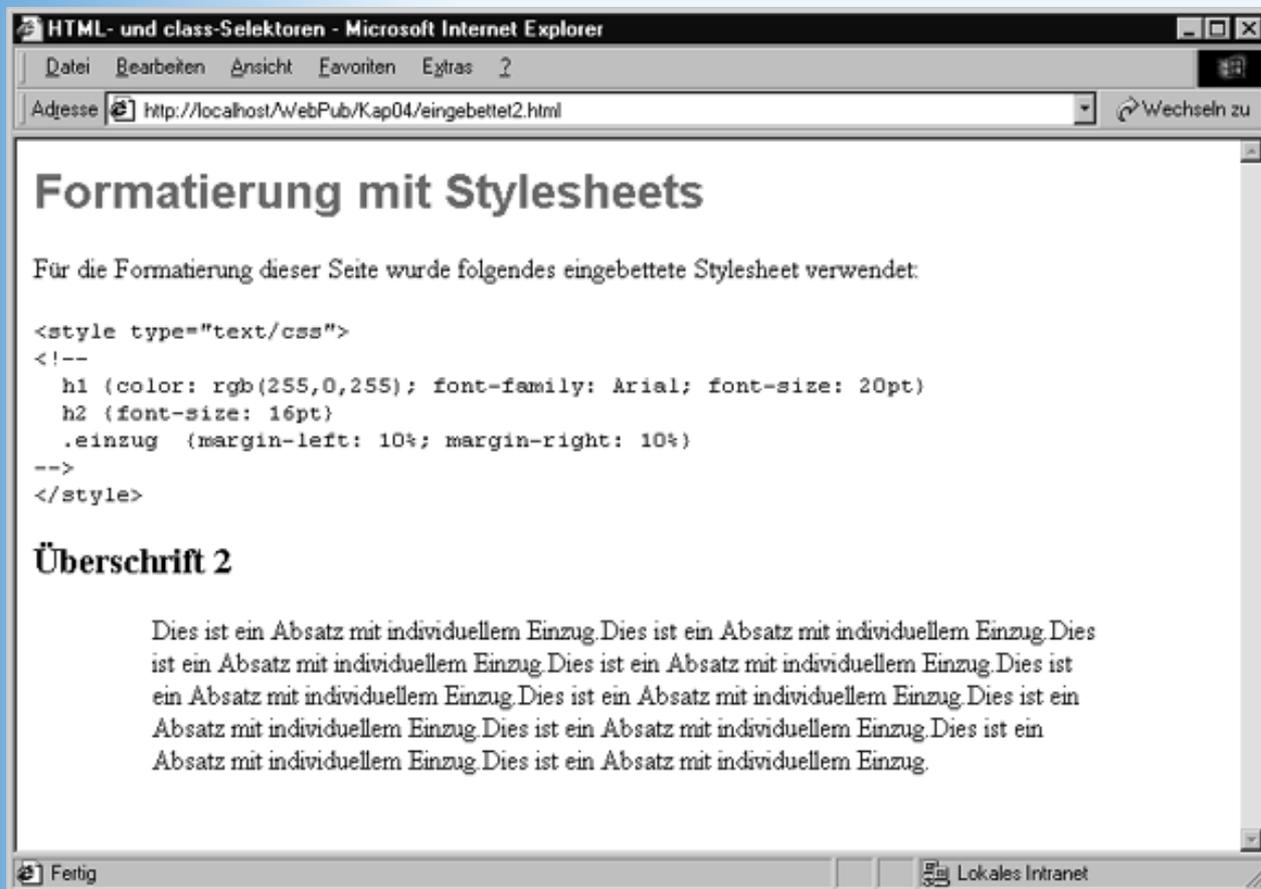


Abbildung 4.6: class-Selektor

ID-Selektoren

Der letzte Typ von Selektor ist der ID-Selektor.



Ein ID-Selektor ist ein beliebiger Name zur Bezeichnung einer Stildefinition, dem ein #-Zeichen vorangeht.

Ein Stil, der mit einem ID-Selektor verbunden ist, wird dem HTML-Element im BODY- Abschnitt zugewiesen, das den gleichnamigen ID-Bezeichner trägt. Wenn Sie also in Ihrem Stylesheet folgenden Stil definieren:

```
#demoabs { margin-left: 10%; margin-right: 10%}
```

wird dieser Stil zur Formatierung des HTML-Elements herangezogen, dessen id-Attribut den Wert "demoabs" hat:

```
<p id="demoabs">
```

Da es in einem HTML-Dokument keine zwei HTML-Elemente mit gleichlautenden id- Bezeichnern geben sollte, wird ein ID-Stil immer nur mit einem HTML-Element verbunden. Der ID-Selektor ist somit der spezifischste der Selektoren.

Selektor	Beschreibung
HTML-Selektor	Ein Stil für alle HTML-Elemente eines Tags
class-Selektor	Ein Stil für mehrere HTML-Elemente (auch verschiedener Tags)
ID-Selektor	Ein Stil für ein HTML-Element

Tabelle 4.1: Spezifität von Selektoren



ID-Selektoren unterscheiden sich von den Inline-Stilen dadurch, dass Sie in einem eingebetteten oder externen Stylesheet definiert werden können und auch relative Formatierungsbefehle verwenden können. Beide Verfahren, ID-Selektoren wie Inline-Stile, dienen dazu, einzelne Elemente individuell zu formatieren, und sollten eher sparsam verwendet werden.

4.3 Die einzelnen Stileigenschaften

Nachdem wir uns nun ausführlich darüber unterhalten haben, wie man Stylesheets aufsetzt und zuweist, sollten wir es uns nicht nehmen lassen, noch einmal auf einen kleinen Streifzug durch die verschiedenen verfügbaren Stileigenschaften zu gehen. Die wichtigsten Eigenschaften haben wir ja bereits in Kapitel 2.8 kennen gelernt. Aber ich bin sicher, dass da noch einige Fragen offen geblieben sind.

Farben und Hintergründe

Mit Farben und Hintergründen haben wir uns bereits in Kapitel 2.8 ausführlich beschäftigt. Damit Sie nicht zurückblättern müssen, wiederholen wir kurz das Wichtigste (und tragen das eine oder andere nach).

Die wichtigsten Eigenschaften für Farben und Hintergrund lauten:

Eigenschaft	Bedeutung
color	Vordergrundfarbe Werte: <Farbe>
background-color	Hintergrundfarbe Werte: <Farbe> transparent
background-image	Hintergrundbild Werte: <uri>
background-repeat	Wie soll die Anzeigefläche des Browser-Fensters mit dem Hintergrundbild ausgefüllt werden Werte: repeat repeat-x repeat-y no-repeat

background-attachment	Soll das Hintergrundbild zusammen mit dem Vordergrund gescrollt werden (Standardeinstellung) oder soll es fixiert bleiben? Werte: scroll fixed
background-position	Positionierung des Hintergrundbildes. Angegeben werden immer ein Wert für die horizontale und ein Wert für die vertikale Positionierung, entweder als Prozentwert, als absolute Längenangabe oder als Schlüsselwort. Werte: [<percentage> <length>]{1,2} [left center right] [top center bottom] Die Positionierung durch Prozentwerte ist ein wenig ungewöhnlich, 0% 0% bedeutet, dass die linke obere Ecke des Bildes in die linke obere Ecke der Hintergrundbox des Elements kommt (entspricht: left top). 10% 20% bedeutet, dass das Bild so verrückt wird, dass der Bildpunkt, der 10% rechts und 20% unter der linken oberen Bildecke liegt, auf die Position 10% rechts und 20% unter der linken oberen Ecke der Hintergrundbox zu liegen kommt.

Tabelle 4.2: Stileigenschaften für Farben und Hintergründe

Farbwerte werden als RGB-Werte

```
color: #ff0000
color: rgb(255,0,0)
color: rgb(100%, 0%, 0%)
```

oder als eines der folgenden Schlüsselwörter angegeben: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow.

Hintergrundbilder

Über Hintergrundbilder als Seitenhintergrund haben wir schon in Kapitel 2.8 einiges erzählt. Die Stileigenschaften background-image, background-repeat, background-attachment und background-position stehen aber für alle Tags im BODY-Bereich zur Verfügung. Das bedeutet, Sie können praktisch für jedes HTML-Element Ihrer Webseite ein eigenes Hintergrundbild vorsehen. So erzeugt beispielsweise der folgende HTML-Code die Webseite aus Abbildung 4.7.

Listing 4.5: hintergrundbild.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Hintergrundbilder</title>
<style type="text/css">
p { margin-left: 10em; margin-right: 10em; text-align: justify; }
</style>
</head>
<body>
<h2>Absatz mit Hintergrundbild</h2>
<p style="background-image: url('saturn.jpg'); background-position: 50% 50%;
background-repeat: no-repeat">Dies ist ein Absatz. Dies ist ein Absatz. ...<p>
```

</body>
</html>



Abbildung 4.7: Hintergrundbild zu einem Absatz.

Ränder und Rahmen

Im vorangehenden Abschnitt haben Sie gesehen, wie man den Hintergrund eines HTML- Elements mit einer Farbe oder einem Bild ausfüllen kann. Was aber ist eigentlich genau der »Hintergrund« eines HTML- Elements? Um diese Frage beantworten zu können, müssen wir ein wenig ausholen.

CSS unterscheidet wie HTML zwischen Block- und Inline-Elementen. Blockelemente sind grob gesagt, alle HTML-Elemente, die in einer neuen Zeile anfangen. <p>, , <h1> bis <h6>, <div> sind Beispiele für Blockelemente. Inline-Elemente sind Elemente, die in der gleichen Zeile (»in line«) liegen können wie der umgebende Inhalt (üblicherweise der Inhalt des übergeordneten Blockelements). , <cite> oder sind Beispiele für Inline-Elemente.

Jedes Blockelement liegt aus Sicht des Browsers in einer Box (Kasten). Diese Box umschließt

- den Inhalt des HTML-Elements
- einen optionalen Freiraum (Padding oder Füllung genannt)
- einen optionalen Rahmen
- einen optionalen Rand.

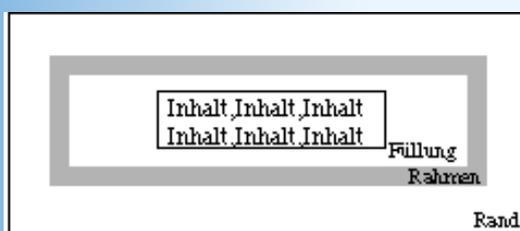


Abbildung 4.8: Aufbau von HTML-Boxen

Füllung (padding), Rahmen (border) und Rand (margin) können über eigene Stileigenschaften eingestellt werden.

Füllung (padding)

Indem Sie Füllraum vorgeben, können Sie den Inhalt eines Elements durch Leerraum von den umliegenden Elementen absetzen. Natürlich könnte man dazu auch einen Rand definieren, doch der Füllraum hat einige Vorteile:

- er liegt innerhalb des Rahmens
- er gehört zum Hintergrund des Elements

Damit hätten wir auch schon die Frage aus der Einleitung zu diesem Abschnitt beantwortet. Der Hintergrund eines Elements umschließt den Inhalt und den Füllraum des Elements (reicht also bis zum Rahmen).



Der Netscape Navigator 4.x lässt immer einen kleinen Streifen zwischen Hintergrund (Inhalt und Füllung) und Rahmen (was auffällt, wenn man einen sichtbaren Rahmen definiert).

Zur Festlegung des Füllraums gibt es fünf Stileigenschaften:

Eigenschaft	Bedeutung
padding-top	Füllraum oben
padding-right	Füllraum rechts
padding-bottom	Füllraum unten
padding-left	Füllraum links
padding	Füllraum zu allen Seiten

Tabelle 4.3: padding-Eigenschaften

Sie brauchen der Eigenschaft nur die gewünschte Größenangabe in Einheiten von %, em, px, ex, pt, mm, cm, in oder pc zuzuweisen, beispielsweise:

```
padding-left: 3em
```

Über die Eigenschaft padding können Sie in einem Zug allen vier Seiten Werte zuweisen:

```
padding: 1em 4em 1em 4em
```

Diese werden der Reihe nach den Füllräumen oben, rechts, unten und links zugewiesen. Wenn Sie nur einen Wert angeben, gilt dieser für alle vier Seiten, wenn Sie nur zwei Werte vorgeben, gilt jeder dieser Werte für die eigene und die gegenüberliegende Seite. Obige Formatierung hätte man also auch wie folgt schreiben können:

padding: 1em 4em

In Abbildung 4.10 können Sie sehen, wie sich die Angabe eines Füllraums auswirkt.

Größenangaben in Stileigenschaften

Viele Stileigenschaften erwarten als Werte Größenangaben. Manchmal kann man diese Angaben in Prozent machen (wie im Falle des Füllraums), ansonsten stehen verschiedene absolute Maßeinheiten zur Verfügung:

em, entspricht der Punktgröße der verwendeten Schrift (2em bedeutet also doppelte Punktgröße der Schrift)

- px, Pixel

- ex, Höhe des Buchstaben x in der umliegenden Schrift

- pt, Punktgröße (unterschiedlich auf den verschiedenen Plattformen)

- mm

- cm

- in, Inch

- pc, Pica

Grundsätzlich sollten Sie relative Größenangaben bevorzugen, also % (Prozent, falls erlaubt) oder em, ex oder px.



Vergessen Sie in Stylesheets nie die Angabe der Maßeinheit. Manche Browser wie der Internet Explorer gehen zwar bei fehlenden Maßeinheiten von Pixelwerten aus, andere Browser aber ignorieren in so einem Fall die ganze Stileigenschaft.

Rahmen (border)

Zur Festlegung des Rahmens gibt es eine ganze Reihe von Stileigenschaften:

Eigenschaft	Bedeutung
border-width	Rahmenbreite Werte: [thin medium thick <length>]{1,4} Diese Eigenschaft ist eine Abkürzung für die vier Eigenschaften: border-top-width, border-right-width, border-bottom-width und border-left-width (siehe padding).
border-color	Farbe des Rahmens
border-style	Rahmenmuster Werte: none dotted dashed solid double groove ridge inset outset

border-top	Zusammenfassung der Eigenschaften für den oberen Rand Werte: <border-top-width> <border-style> <color>
border-right	Zusammenfassung der Eigenschaften für den rechten Rand Werte: <border-top-width> <border-style> <color>
border-bottom	Zusammenfassung der Eigenschaften für den unteren Rand Werte: <border-top-width> <border-style> <color>
border-left	Zusammenfassung der Eigenschaften für den linken Rand Werte: <border-top-width> <border-style> <color>
border	Abkürzung für die vier Eigenschaften: border-top, border-right, border-bottom und border-left (siehe padding).

Tabelle 4.4: border-Eigenschaften

Um ein Element einzurahmen, müssen Sie zumindest eine Rahmenbreite und einen Rahmenstil vorgeben:

```
border-width: 3px; border-style: solid
```

Ein Rahmen muss nicht immer ein Rahmen sein. Sie können den Rahmen auch verwenden, um ein Element zu »unterstreichen« oder es durch Balken oben und unten hervorzuheben:

```
<style type="text/css">
  h2 { border-width: 0.5em 0em 0.5em 0em;
        border-style: solid; border-color: maroon }
</style>
```



Wegen des Internet Explorers müssen Sie beim Zeichnen einzelner Rahmenseiten, die nicht zu zeichnenden Rahmenseiten explizit auf 0 setzen.

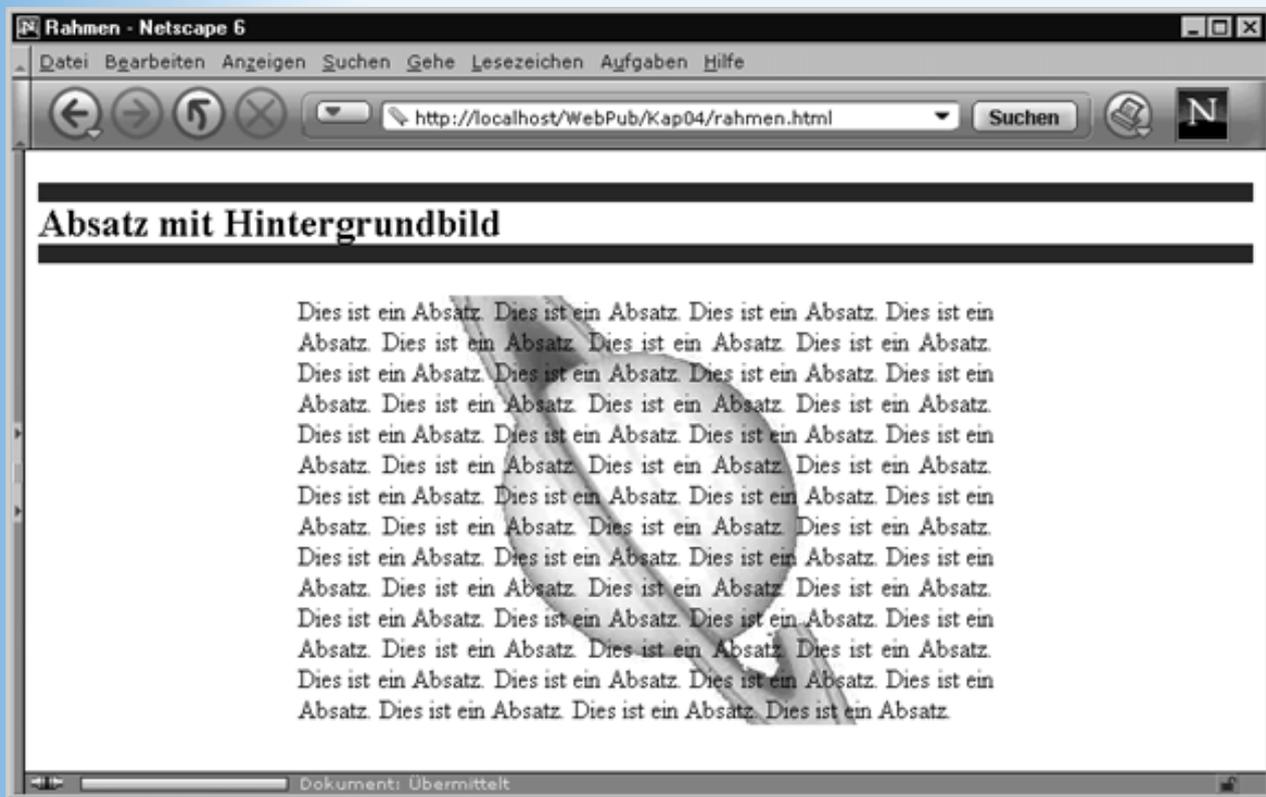


Abbildung 4.9: Rahmen um h2-Überschrift

Rand (margin)

Für den Rahmen gibt es wie für den Füllraum fünf Stileigenschaften. Eine Eigenschaft für jede Seite und eine Stileigenschaft für alle Seiten zusammen.

Eigenschaft	Bedeutung
margin-top	Rand oben
margin-right	Rand rechts
margin-bottom	Rand unten
margin-left	Rand links
margin	Rand zu allen Seiten

Tabelle 4.5: margin-Eigenschaften

In Abbildung 4.10 können Sie sehen, wie sich die Angabe eines Füllraums auswirkt.

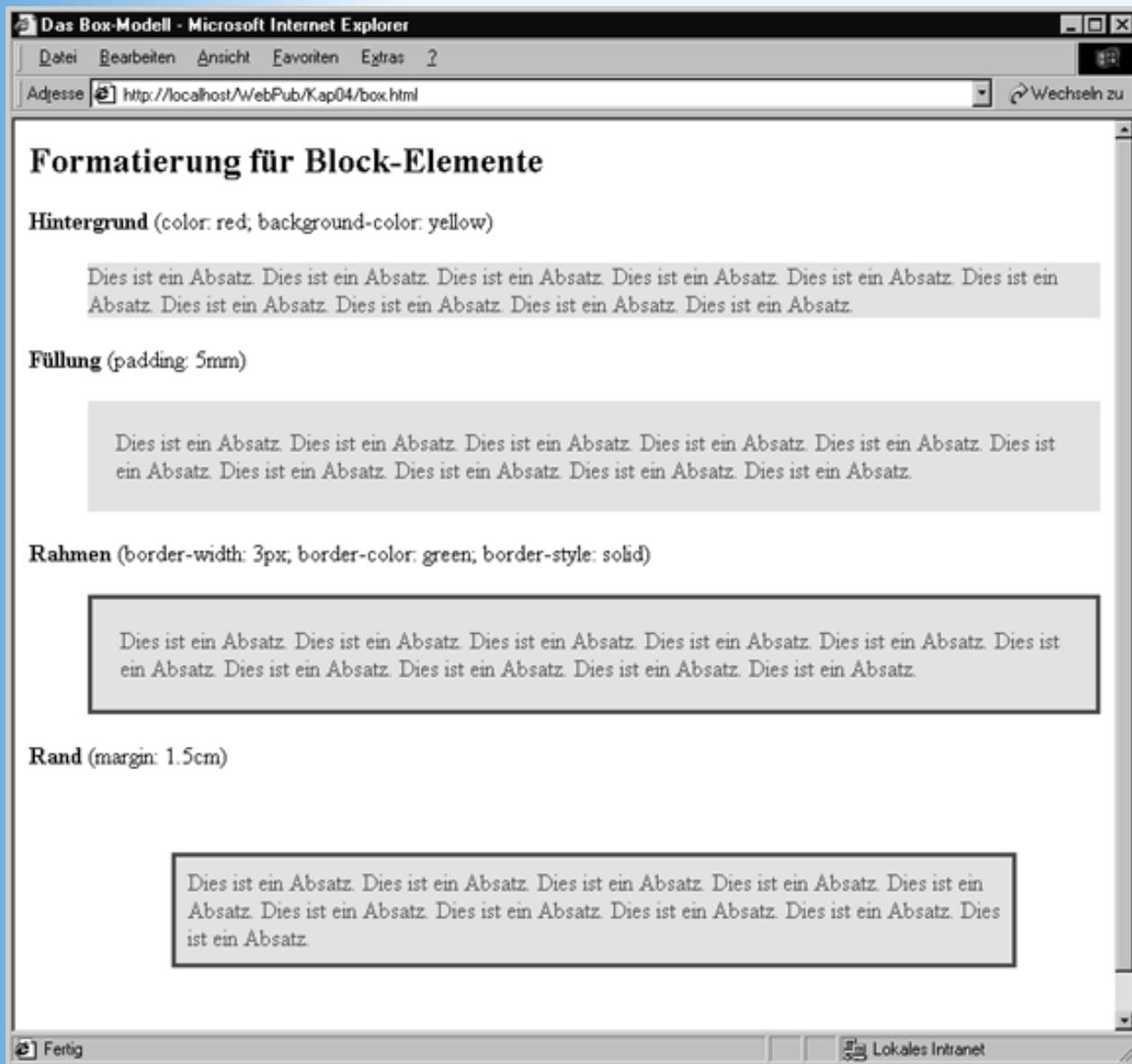


Abbildung 4.10: Hintergrund, Füllung, Rahmen und Rand



Die Eigenschaften für Hintergrund, Füllung, Rahmen und Rand können auch auf Inline-Elemente angewendet werden, führen dort aber unter Umständen zu ungewöhnlichen Ergebnissen.

Texteigenschaften

Die Texteeigenschaften zerfallen in zwei große Blöcke:

- die Schrifteigenschaften
- die Freiraumeigenschaften

Schriften

Über den Einsatz und die Formatierung von Schriften haben wir uns bereits in Kapitel 2.8 befasst. Hier daher nur eine kurze Zusammenfassung:

Eigenschaft	Bedeutung
font-family	Schriftfamilie Werte: Geordnete Liste von Schriftnamen, beispielsweise: font-family: 'Mein Arial', Arial, sans-serif Beachten Sie, dass am Ende der Reihe immer einer der folgenden Werte stehen sollte: serif, sans-serif, cursive, fantasy, monospace (siehe Kapitel 2.8.3).
font-style	Schriftstil Werte: normal italic oblique
font-weight	Schriftbreite Werte: lighter normal bold bolder 100 200 300 400 500 600 700 800 900
font-size	Schriftgröße Werte: <Größenangabe in em, pt, px ,ex oder %> larger smaller
text-decoration	Textauszeichnung Werte: none [underline overline line-through blink]
text-indent	Texteinzug (nur für Blockelemente) Werte: <Größenangabe in em, pt, px ,ex, cm, mm oder %>
text-shadow	Werte: none [<color> <length> <length> <length>? ,]* [<color> <length> <length> <length>?]
text-transform	Textumwandlungen Werte: capitalize uppercase lowercase none

Tabelle 4.6: Schrifteigenschaften

Freiräume in Textzügen

Die Eigenschaften padding und margin erzeugen einen Freiraum um den Inhalt eines gegebenen Elements. Eine andere Frage ist, wie man den Freiraum innerhalb des Textes organisiert. Auch hierfür gibt es eigene Stileigenschaften

Eigenschaft	Bedeutung
text-align	Textausrichtung (nur für Blockelemente) Werte: left right center justify

vertical-align	vertikale Textausrichtung (nur für Inline-Elemente) Werte: baseline sub super top text-top middle bottom text-bottom % (der Linienhöhe)
text-indent	Texteinzug (nur für Blockelemente) Werte: <Größenangabe in em, pt, px ,ex, cm, mm oder %>
line-height	Zeilenhöhe (von einer Basislinie zur nächsten) Werte: <Größenangabe in em, pt, px ,ex, cm, mm oder % oder als reine Zahl> Reine Zahlen oder Prozentangaben beziehen sich immer auf die Punktgröße der betreffenden Schriftart. Ist diese beispielsweise gleich 10 pt und geben Sie als Wert für line-height 1.5 an, ist die Linienhöhe danach 15 pt.
word-spacing	Zusätzlicher Freiraum zwischen Wörtern eines Absatzes Werte: <Größenangabe in em, pt, px ,ex, cm, mm>
letter-spacing	Zusätzlicher Freiraum zwischen Buchstaben eines Wortes Werte: <Größenangabe in em, pt, px ,ex, cm, mm>

Tabelle 4.7: Texteigenschaften

Die Eigenschaften word-spacing und letter-spacing sollten nur sparsam zur Hervorhebung einzelner Textpassagen genutzt werden (beispielsweise gesperrte Wörter). Ansonsten sollte man immer daran bedenken, dass die Font-Designer sich bei der Entwicklung ihrer Schriften bereits intensiv mit der Frage der optimalen Buchstaben und Wortabstände gemacht haben.

Auch die Eigenschaft text-indent sollte uns eine Anmerkung wert sein.

Wenn Sie zufällig über die Faber and Faber-Ausgabe von »Lord of the Flies« verfügen (ISBN: 0-571-08483-4), so schlagen Sie diese doch bitte einmal auf. In diesem Buch wird jede erste Zeile eines Absatzes leicht eingerückt - keine ganz ungewöhnliche, aber auch nicht sonderlich aufregende Formatierung, oder?

For
MY MOTHER AND FATHER

CHAPTER ONE

The Sound of the Shell

THE BOY with fair hair lowered himself down the last few feet of rock and began to pick his way towards the lagoon. Though he had taken off his school sweater and trailed it now from one hand, his grey shirt stuck to him and his hair was plastered to his forehead. All round him the long scar smashed into the jungle was a bath of heat. He was clambering heavily among the creepers and broken trunks when a bird, a vision of red and yellow, flashed upwards with a witch-like cry; and this cry was echoed by another.

"Hi!" it said, "wait a minute!"

The undergrowth at the side of the scar was shaken and a multitude of raindrops fell pattering.

"Wait a minute," the voice said, "I got caught up."

The fair boy stopped and jerked his stockings with an automatic gesture that made the jungle seem for a moment like the Home Counties.

The voice spoke again.

"I can't hardly move with all these creeper things."

The owner of the voice came backing out of the undergrowth so that twigs scratched on a greasy wind-breaker. The naked crooks of his knees were plump, caught and scratched by thorns. He bent down, removed the thorns carefully, and turned round. He was shorter than the fair boy and very fat. He came forward, searching out safe lodgements for his feet, and then looked up through thick spectacles.

"Where's the man with the megaphone?"

Abbildung 4.11: Englische Ausgabe von *Lord of the Flies*

Nicht aufregend? Von wegen! Das Einrücken von Textzeilen hat den Webdesignern weltweit schon einiges an Kopfzerbrechen bereitet. Einige versuchten es mit Leerzeichen, , die Sie in `<pre>`-Tags fassten, damit sie nicht vom Browser ignoriert werden, andere erzeugten blinde GIF-Bilder und kämpften damit, diese an die Größe der umliegenden Schrift anzupassen.

Blinde GIFs

Blinde GIFs sind 1x1-Pixelgrafiken, die meist einen weißen oder transparenten Punkt enthalten und in vielen Designs als Platzhalter eingesetzt werden. Über die `width`- und `height`-Attribute des `img`-Tags wird das blinde GIF so ausgedehnt, dass es den freizuhaltenden Platz einnimmt.

Dank der Stileigenschaft `text-indent` sind solche Tricks - zumindest soweit es das Einziehen von Anfangszeilen angeht - überflüssig geworden. In Listing 4.6 sehen Sie HTML-Code, der das Layout des Faber-Buches nachstellt.

Listing 4.6: `textindent.html` - Absätze mit eingerückten Anfangszeilen

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Texteinzug</title>
  <style type="text/css">
    body { margin-left: 1em; margin-right: 10em }
```

```

h1 { text-align: center;
      font-family: serif; font-style: italic; font-size: 20 pt }
h1.chapter { font-family: 'Courier', monospace; font-size: 12 pt;
              font-style: normal}
p { text-align: justify; text-indent: 1em }
</style>
</head>
<body>
<table cols="3" width="100%">
<colgroup span="3">
  <col width="10%">
  <col width="400">
  <col width="*">
</colgroup>
<tr>
  <td width="10%"></td>
  <td width="400">

<h1 class="chapter">CHAPTER ONE</h1>
<h1>The Sound of the Shell</h1>
<p>The Boy with fair hair lowered himself down the last few feet of rock and
began to pick his way towards the lagoon. Though he had taken off his school
sweater and trailed it now from one hand, his great shirt struck to him and his
hair was plastered to his forehead. All round him the long scar smashed into
the jungle was a bath of heat. He was clambering heavily among the creepers and
broken trunks when a bird, a vision of red and yellow, flashed upwards with a
witch-like cry; and this cry was echoed by another.</p>
<p>"Hi!" it said, "wait a minute!"</p>
<p>The undergrowth at the side of the scar was shaken and a multitude of
raindrops fell pattering.</p>
<p>"Wait a minute," the voice said, "I got caught up."</p>
      </td>
  <td width="*"></td>
</tr>
</table>
</body>
</html>

```



Beachten Sie, dass wir die Breitenangaben für die einzelnen Spalten in den <td>-Deklarationen wiederholt haben. Dies ist derzeit noch notwendig, weil der Netscape Navigator 4.x (im Gegensatz zur 6er-Version oder zum Internet Explorer) die Breitenangaben in der <colgroup>-Deklaration nicht auswertet.

Listen

Einige Stileigenschaften wurden speziell zur Formatierung von Listen definiert.

Eigenschaft	Bedeutung
list-style-type	Welches Listensymbol soll verwendet werden Werte: disc circle square decimal lower-roman upper-roman lower-alpha upper-alpha none
list-style-image	Statt eines der vordefinierten Listensymbole kann man auch den URL zu einer Grafikdatei mit einem passenden Symbol angeben, beispielsweise: ul { list-style-image: url(ellipse.png) }
list-style-position	Position des Listensymbols in Relation zum Text des Listenelements Werte: inside outside
list-style	Zusammenfassung für list-style-type, list-style-position und list-style-image

Tabelle 4.8: Listeneigenschaften



Wenn Sie ausgefallenerere Grafiken als Listensymbole verwenden wollen, die mehr Platz einnehmen als die üblichen Punkte oder Zahlen, ist es meist besser, die Liste durch eine Tabelle zu simulieren und die »Listensymbole« mittels in die erste Spalte und die Listenelemente in die zweite Spalte einzutragen (siehe Abbildung 4.12). Diese Technik hat im Übrigen auch den Vorteil, dass sie von praktisch allen Browsern unterstützt wird.

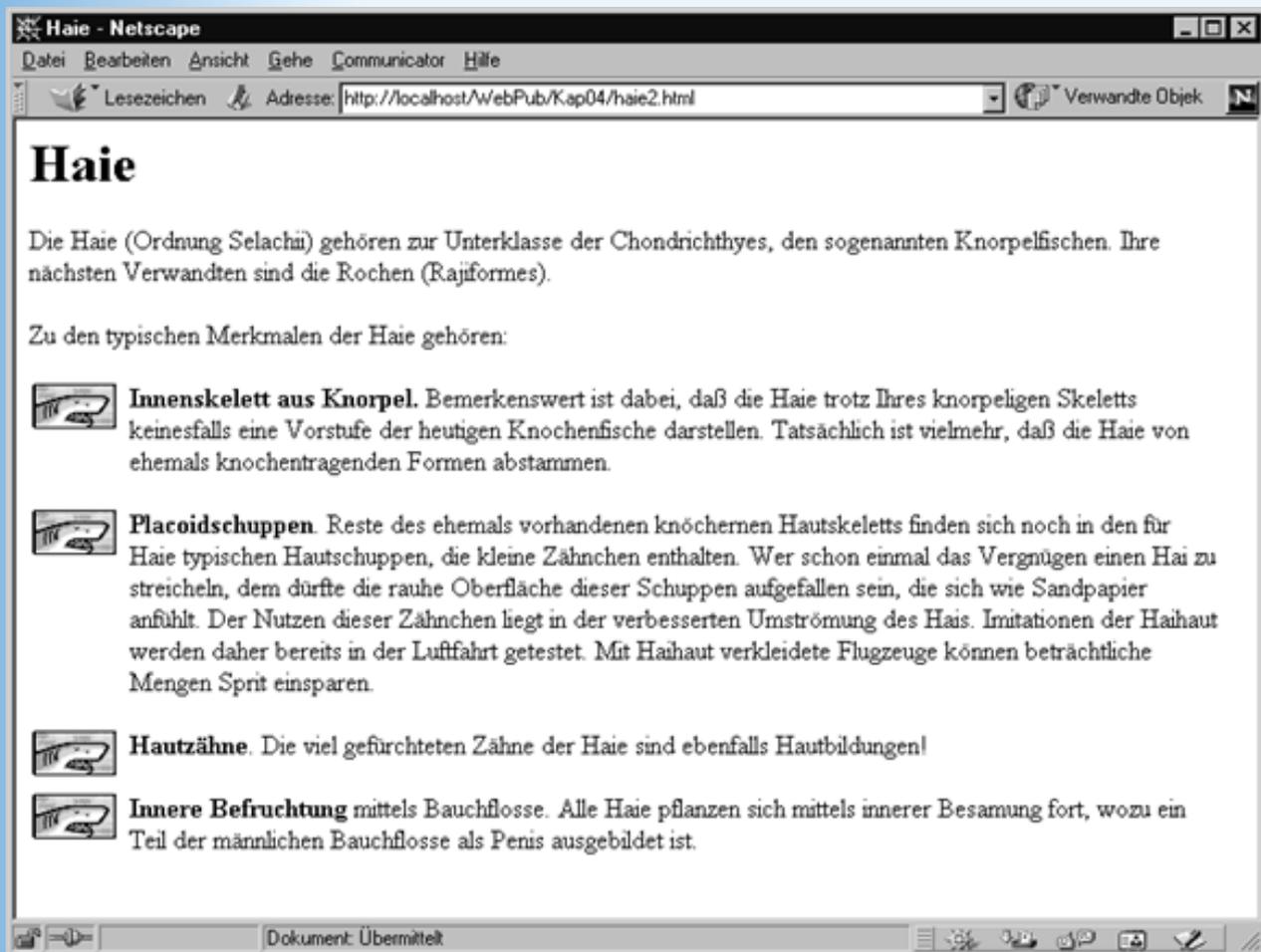


Abbildung 4.12: Durch eine Tabelle simulierte Liste

Tabellen

Seit der CSS2-Spezifikation gibt es eine Reihe von Stileigenschaften, die speziell für die Formatierung von Tabellen definiert wurden.

Eigenschaft	Bedeutung
caption-side	Wo soll die Tabellenüberschrift (caption) in Relation zur Tabelle angezeigt werden? Werte: top bottom left right
table-layout	Wie soll die Tabelle erzeugt werden? Dabei geht es vor allem darum, ob die Breite der Spalten an den Inhalt der Zellen angepasst werden soll (auto) oder allein aus den width-Angaben berechnet werden soll (fixed). Werte: auto fixed
border-collapse	Wie sind die Rahmen benachbarter Zellen zu behandeln. Fallen Sie zusammen (collapse) oder sind sie als einzelne Rahmenelemente zu betrachten (separate). In letzterem Fall kann man mit border-spacing den Abstand zwischen den Rahmenelementen festlegen. Werte: collapse separate

Tabelle 4.9: Tabelleneigenschaften

Sonstige Stile

Eine vollständige Liste aller aktuell definierten Stile finden Sie stets auf der Website des W3C-Konsortiums: <http://www.w3.org>. Halten Sie dort nach den Standards zu CSS1 und CSS2 Ausschau.

4.4 Stylesheets definieren und zuweisen II

Zum Schluss dieses Kapitels müssen wir noch einmal etwas technischer werden. Vor allem sollen Sie endlich erfahren, wie man Stylesheets in externen Dateien definiert und diese externe Stildefinitionen in Webseiten aufnimmt, denn dies ist die professionellste und in Hinblick auf die Zukunft von HTML (die in Richtung XHTML und XML weist) auch die sicherste Form der Einbindung von Stylesheets.

Externe Stylesheets

Statt Ihre Stile in Form eines eingebetteten Stylesheets direkt auf den Seiten abzuspeichern, können Sie diese auch in externen Dateien mit der Dateierdung .css abspeichern und mit Hilfe eines Links in Ihre Webseiten einbinden. Der Vorteil dieses Verfahrens liegt auf der Hand: ein einmal extern definiertes Stylesheet kann durch einfache Verknüpfung oder Import in beliebige Webseiten aufgenommen werden.



Externe Stylesheets arbeiten auch fehlerfrei mit XHTML-Dokumenten zusammen. Wenn Sie Ihre Webseiten also nach dem XHTML-Standard aufsetzen, sollten Sie konsequent mit externen Stylesheets arbeiten (siehe Anmerkung in Abschnitt 4.2.2, »Ältere und neuere Browser«).

Externe Stylesheets werden ganz genauso wie eingebettete Stylesheets aufgesetzt, nur dass Sie nicht innerhalb des Header-Teils eines Webdokuments, sondern in einer eigenen Datei definiert werden.

Externes Stylesheet erzeugen

1. Legen Sie mit einem Texteditor eine neue Datei an.
2. Geben Sie Ihre Stylesheet-Definition ein, beispielsweise

Listing 4.7: externsheet.css

```
h1 { color: rgb(255,0,255); font-family: Arial; font-size: 20pt }
h2 { font-size: 16pt }
.einzug { margin-left: 10%; margin-right: 10%; text-align: justify
```

3. Speichern Sie die Datei mit der Extension .css.

Externe Stylesheets in eine Webseite aufnehmen

Um Stildefinitionen aus einem externen Stylesheet zu verwenden, gibt es zwei Möglichkeiten:

- Sie können das externe Stylesheet mit der Webseite verknüpfen. Dies ist der übliche Weg.

```
<link rel=StyleSheet type="text/css" href="externsheet.css" />
```

- Sie können das externe Stylesheet in ein eingebettetes Stylesheet der Seite importieren. Hierzu verwendet man das Schlüsselwort @IMPORT gefolgt von dem relativen oder absoluten URL der externen Stylesheet-Datei:

```
<style>
<!--
  @import url(externessheet.css);
  #einzug { margin-left: 10%; margin-right: 10%}
-->
</style>
```



Der Navigator 4 unterstützt das Verknüpfen mit einem externen Stylesheets (<link>), aber nicht die @import-Syntax.

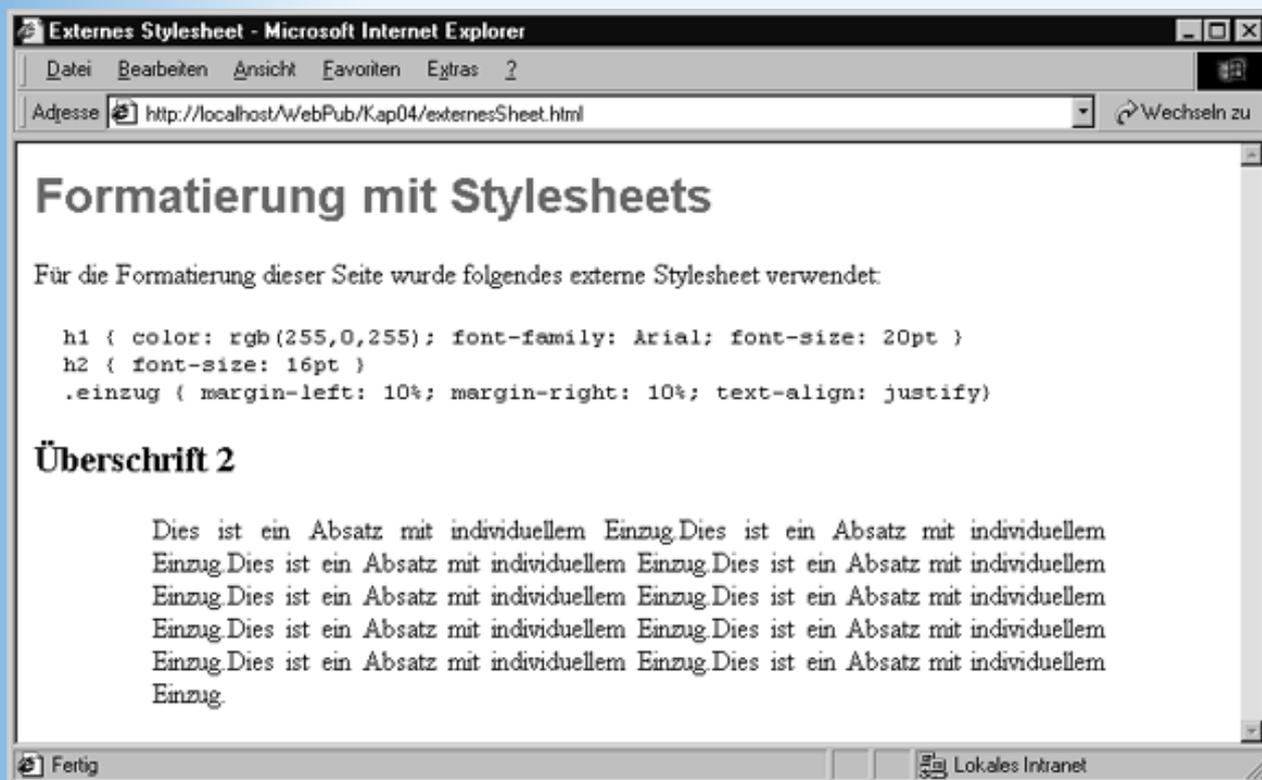


Abbildung 4.13: Formatierung mit einem externen Stylesheet

Kombinierte Selektoren

In Abschnitt 4.2.3 haben Sie drei Arten von Selektoren kennen gelernt: HTML- Selektoren, class-Selektoren und ID-Selektoren. Durch Kombination dieser elementaren Selektoren können Sie neue spezifischere Selektoren bilden. CSS kennt zwei Formen von kombinierten Selektoren:

- Kombinationen von HTML-Selektor und class-Selektor
- Kombinationen von übergeordneten und untergeordneten HTML-Selektoren

class-Selektoren auf einzelne Tags beschränken

Stellen Sie sich vor, Sie haben in Ihrem Webdokument Absätze, die Sie unterschiedlich formatieren wollen. Sie können dies lösen, indem Sie einen Standardstil für alle <p>- Absätze und einen class-Stil für die abweichend zu formatierenden Absätze definieren:

```
p          { font-family: Arial; margin-left: 10%; margin-right: 10%}  
.einzug   { margin-left: 25%; margin-right: 25%}
```

Diese Vorgehensweise ist zwar absolut korrekt und führt auch zu dem gewünschten Ergebnis, dennoch würde man diese Lösung als »unsauber« bezeichnen. Wieso? Auf der einen Seite wollen wir in diesem Beispiel einen Stil (einzug) definieren, den wir speziell zur Formatierung von Absätzen verwenden wollen. Auf der anderen Seite wird dies aus der Stildefinition nicht ersichtlich und es hindert uns nichts, den gleichen Stil auch auf Bilder oder Tabellen anwenden. Man kann dem abhelfen, indem man dem class-Selektor den HTML-Selektor des Tags, auf das sich der class-Stil beziehen soll, voranstellt:

```
p          { font-family: Arial; margin-left: 10%; margin-right: 10%}  
p.einzug   { margin-left: 25%; margin-right: 25%}
```

Jetzt haben wir einen Standardstil für alle <p>-Absätze und einen speziellen class-Stil für alle <p>-Absätze, denen wir das class-Attribut class="einzug" zuweisen. Aus der Stildefinition wird nun für den Webdesigner ersichtlich, dass der Stil p.einzug nur für <p>- Absätze gedacht ist, und der Webbrowser stellt sicher, dass die Attributdefinition class="einzug" für andere HTML-Tags ignoriert wird.

Vielleicht mag Ihnen die Begrenzung von class-Stilen auf einzelne HTML-Tags unnötig oder pedantisch vorkommen, aber Sie fördert den strukturierten Aufbau von Stylesheet- Definitionen und hilft Ihnen bei späteren Überarbeitungen, Ihre Stylesheet-Definition besser zu verstehen.



Achten Sie darauf, dass kein Leerzeichen zwischen dem Tag-Selektor und dem class-Selektor steht! Ansonsten definieren Sie nämlich einen Kontext (siehe unten).

Stile für einen bestimmten Kontext definieren

Durch die Kombination Tag.Klassenname beschränken Sie einen class-Stil auf die Verwendung mit einem bestimmten Tag. Eine andere Form der Begrenzung ist die Angabe des umliegenden Kontextes. Ein solcher Kontext kann ein HTML-Tag, ein class- Stil oder ein ID-Stil sein.

Nehmen wir an, Sie definieren einen Stil zur farblichen Hervorhebung von Textpassagen:

```
.hervor { color: red}
```

Wo immer Sie im BODY-Abschnitt Textpassagen hervorheben wollen, fassen Sie diese in -Tags und weisen die Klasse .hervor zu:

```
<span class="hervor">Dies ist die Textpassage</span>.
```

Nehmen wir nun weiter an, Sie definieren auch in einer Ihrer Überschriften eine Hervorhebung:

```
<h1>Formatierung mit <span class="hervor">Stylesheets</span></h1>
```

An sich ist das kein Problem, doch würden Sie gerne alle <h1>-Überschriften rot färben. Dies würde aber wiederum bedeuten, dass die Hervorhebung in den <h1>-Überschriften verloren geht. Statt nun eine zweite Klasse .hervor2 zu definieren, können wir auch vorgeben, dass die Klasse .hervor, wenn Sie in <h1>-Überschriften verwendet wird, mit der Farbe Blau verbunden werden soll.

```
h1 .hervor { color: blue}
```

Beachten Sie, dass hier kein Komma zwischen h1 und .hervor steht. Der Selektor bezieht sich jetzt auf alle HTML-Elemente, denen die Klasse .hervor zugewiesen wurde und die innerhalb eines <h1>-Elements liegen.

Listing 4.8: eingebettet3.html - Kontextsensitive Stile

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Selektoren und Kontext</title>
  <style type="text/css">
    <!--
    h1 { color: red; font-family: Arial; font-size: 20pt }
    h2 { font-size: 16pt }
    h1 .hervor { color: blue}
    p .hervor { color: red}
    -->
  </style>
</head>
<body>
<h1>Formatierung mit <span class="hervor">Stylesheets</span></h1>
...
<h2>Überschrift 2</h2>
<p class="einzug">Dies ist ein Absatz mit individuellem Einzug.<span
class="hervor">Dies ist ein Absatz mit individuellem Einzug</span>.Dies ist ein
Absatz mit individuellem Einzug.Dies ist ein Absatz mit individuellem
Einzug.Dies ist ein Absatz mit individuellem Einzug.Dies ist ein Absatz
mitindividuellem Einzug.Dies ist ein Absatz mit individuellem Einzug.Dies ist
ein Absatz mit individuellem Einzug.Dies ist ein Absatz mit individuellem
Einzug.Dies ist ein Absatz mit
individuellem Einzug.</p>
</body>
</html>
```

Da die farblichen Hervorhebungen bei der Umwandlung in Grautöne verloren gehen, haben wir hier auf eine Abbildung verzichtet. Sie können die HTML-Datei *eingebettet3.html* aber von der Buch-CD in Ihren Browser laden und sich in diesem das Ergebnis anschauen.

Sie können auch mehrere Kontextebenen angeben. Betrachten Sie zum Beispiel folgende Stildefinition:

```
ol ul li {list.style: square }
```

Diese Stildefinition gilt für alle Listenelemente, die in einer ungeordneten Liste liegen, die selbst in einer umliegenden geordneten Liste enthalten ist.

Oder Sie legen fest, dass eine Stildefinition nur für ``-Elemente gilt, die in Absätze liegen, die mit dem Klassenstil `.einzug` formatiert sind:

```
p.einzug em {font-size: larger}
```

Selektoren für Hyperlinks und Blockanfänge

Und was ist mit den Hyperlinks? Wir erinnern uns: der Webbrowser unterscheidet zwischen nicht-besuchten, besuchten und aktivierten Hyperlinks und nutzt üblicherweise verschiedene Farben, um die betreffenden Zustände eines Hyperlinks darzustellen. Nach der alten Formatierung konnte man die Farben für die drei Linkzustände mit den Attributen `link`, `vlink` und `alink` selbst festlegen:

```
<body link="red" vlink="maroon" alink="fuchsia">
...
</body>
```

Wie aber kann man diese Linkfarben mit Hilfe von Stylesheets festlegen. Schließlich gibt es nur ein Anker-Element `<a>` und auch nur eine Stileigenschaft `color` für die Vordergrundfarbe?

Die Lösung liegt in einer kleinen Mogelei. Glücklicherweise müssen aber nicht wir mogeln, das hat schon der CSS-Standard für uns getan. Er hat einfach drei Pseudo-Klassen für die drei Linkzustände definiert:

- `A:link` (für noch nicht besuchte Links)
- `A:visited` (für bereits besuchte Links)
- `A:active` (für aktive Links)

Diese Pseudo-Klassen gelten für alle `<a>`-Elemente mit `href`-Attributen. Die obige Formatierung für die drei Linkzustände könnte in einem Stylesheet also wie folgt aussehen:

```
A:link {color: red}
A:visited {color: maroon}
A:active {color: fuchsia}
```



Es gibt noch weitere Pseudo-Klassen, wie z.B. `:before` und `:after`, die einen Text vor beziehungsweise nach jedem HTML-Element einfügen, doch werden diese derzeit nur vom Netscape 6-Browser unterstützt. Wer mehr über diese Formatierungsmöglichkeiten erfahren möchte, schlage bitte in der CSS2-Spezifikation nach.

4.5 Vererbung und Cascading

Bisher haben wir die Stylesheets meist so betrachtet, als ob sie ganz für sich alleine stünden. Dies ist aber selten der Fall. Vielmehr kommt es häufig zu Überschneidungen oder gar Konflikten zwischen den Stildefinitionen ineinander verschachtelter HTML-Elemente oder durch nacheinander definierte Stildefinitionen.

Vererbung

Innere Elemente übernehmen die Stile umliegender Elemente, behalten aber ihre spezifischen Einstellungen.

Wenn Sie innerhalb eines Textabsatzes (<p>) eine Textpassage als Zitat (<cite>) formatiert haben, haben Sie HTML-Elemente bereits mehrfach ineinander verschachtelt.

Ganz außen stehen die <body>-Tags. Innerhalb der <body>-Tags stehen die <p>-Tags der Absätze und innerhalb eines Absatzes finden sich irgendwo die <cite>-Tags für das Zitat:

```
<body>
<h1>Vererbung von Stilen</h1>
<p>Heraklit sah im Krieg <cite>&quot;... den Vater aller Dinge und den König
aller.&quot; </cite></p>
</body>
```

Wenn Sie diesen HTML-Elementen CSS-Stile zuweisen, werden diese in gleicher Weise verschachtelt wie die HTML-Elemente - mit der Konsequenz, dass jedes innere Element die Formatierung des umliegenden Elements erbt. Dabei ist es unwesentlich, wie den Elementen die Stile zugewiesen wurden (externe Stylesheets, eingebettete Stylesheets, HTML-, class- oder id-Selektor, Inline-Stile). Dass im folgenden Beispiel alle drei Stile über HTML-Selektoren zugeordnet werden, dient also lediglich der besseren Übersichtlichkeit:

```
<style type="text/css">
  body    {color: white; background-color: black}
  p       {color: gray;   font-size: 18pt}
  cite    {color: yellow}
</style>
```

Wie wird der obige Text durch das obige Stylesheet formatiert?

- Der body-Stil legt grundsätzlich fest, dass alle Elemente innerhalb der <body>-Tags Weiß als Textfarbe und Schwarz als Hintergrundfarbe verwenden (also konträr zu den Standardeinstellungen formatiert werden).
- Absätze, die im Rumpfteil der Webseite liegen, übernehmen (erben) diese Formatierung. So übernimmt die h1-Überschrift die weiße Textfarbe.
- Für <p>-Absätze wurde zusätzlich ein eigener Stil vorgesehen. Dieser setzt die Schriftgröße auf 18 Punkte hoch und... definiert eine eigene Textfarbe. Jetzt sind für den <p>-Absatz aber zwei Textfarben definiert: eine im p-Stil und eine im body-Stil. Wie man erwarten würde, setzt sich die Definition im p-Stil durch. Wir können uns also merken:
- Wenn ein innerer Stil (beispielsweise für das <p>-Tag) die gleiche Eigenschaft (beispielsweise color) verwendet wie ein äußerer Stil (beispielsweise für das <body>-Tag), haben die Einstellungen im inneren Stil Vorrang vor den Einstellungen im äußeren Stil. (Anders ausgedrückt: der explizit für das Element spezifizierte Stil hat Vorrang vor dem geerbten Stil.)
- Für das Zitat gilt damit, dass es die Einstellungen für die Hintergrundfarbe und die Schriftgröße von den übergeordneten HTML-Elementen übernimmt, aber in eigener Farbe (Rot) dargestellt wird.



Abbildung 4.14: Vererbung von Stilen



Nicht alle Stileigenschaften werden vererbt. margin, padding und border werden beispielsweise nicht vererbt. Die Hintergrundfarbe (background-color) wird ebenfalls nicht vererbt, scheint aber durch.

Cascading

Konfliktsituationen können auch entstehen, wenn mehrere Stildefinitionen zur Formatierung eines Elements vorliegen. Dabei gilt der gleiche Grundsatz wie bei der Vererbung:

Alle Stildefinitionen, die ein Element oder eine Textpassage betreffen, werden ausgewertet. Für Formatierungen, die in mehreren Stilen vorgesehen werden, muss entschieden werden, welche Einstellung angewendet werden soll.

Hierbei gibt es vier Kriterien zu unterscheiden:

Herkunft

Stildefinitionen können von verschiedenen Orten herkommen.

- Eventuell verwendet der Browser eigene Stylesheets für die Standardformatierung der in ihm angezeigten Webseiten.
- Eventuell erlaubt der Browser den Anwendern eigene Stylesheets zur Standardformatierung der anzuzeigenden Webseiten anzugeben.
- Der Autor der Webseite definiert ein eingebettetes Stylesheet oder einen Link zu einem externen Stylesheet.
- Der Autor der Webseite importiert in einem eingebetteten Stylesheet eine Datei mit Stildefinitionen.

Kommt es in so einem Fall zu Konflikten (etwa weil in allen vier Fällen ein Stil für die Vordergrundfarbe von `<p>`-Absätzen vorgesehen ist), so haben die in der obigen Aufzählung weiter unten aufgeführten Stildefinitionen Vorrang vor den anderen Stildefinitionen. Kurz gesagt, die Stildefinitionen des Autors einer Webseite haben Vorrang vor Standardstildefinitionen, die der Websurfer oder der Browser vorgeben, und

unter den vom Webautor vorgesehenen Stildefinitionen haben die Stile aus importierten Stylesheets die geringste Priorität.

Spezifität

Weisen zwei Stildefinitionen unterschiedliche Spezifität auf, setzt sich der stärker spezifische Stil durch.

Die Spezifität eines Stils lässt sich an seinen Selektoren ablesen. ID-Selektoren sind spezifischer als class-Selektoren und diese sind wiederum spezifischer als HTML- Selektoren:

ID-Selektor > class-Selektor > HTML-Selektoren



Werden kombinierte Selektoren verwendet, müssen Sie die Anzahl der ID-, class- und HTML-Selektoren zählen und vergleichen.

Ausgehend von den folgenden Stildefinitionen:

```
body      {color: white; background-color: black}
p.rot     {color: red }
p         {color: yellow;      font-size: 18pt}
```

werden die Absätze

```
<p>Heraklit sah im Krieg <cite>&quot;... den Vater aller Dinge und den König
aller.&quot;</cite></p>
<p class="rot">Vergil bewertete den Krieg etwas anderes: <cite>&quot;Kein Heil
ist in dem Krieg; um Frieden dich flehen wir alle!&quot;</cite></p>
```

wie folgt formatiert:

Für den ersten Absatz gibt es nur eine Stildefinition (p). Die Stildefinition p.rot spielt hier keine Rolle, weil der erste Absatz nicht zur Klasse .rot gehört. Folglich erscheint der erste Absatz in gelber Schrift.

Für den zweiten Absatz gibt es zwei Stildefinitionen (p und p.rot) mit konkurrierenden Angaben für die Vordergrundfarbe. In diesem Falle setzt sich die Definition color:red durch, weil sie aus dem spezifischeren Stylesheet stammt.



Beachten Sie, dass der Konkurrenzkampf nur einzelne Stileigenschaften betrifft. So wird auch der zweite Absatz in einer Schrift von 18 Punkten angezeigt, weil er diese Eigenschaft aus der Stildefinition p übernimmt.

Definitionsreihenfolge

Weisen zwei konkurrierende Stildefinitionen gleiche Spezifität auf, setzt sich der zuletzt deklarierte Stil durch.

Ein solcher Fall kann beispielsweise eintreten, wenn Sie mehrere externe Stylesheet verwenden.



Abbildung 4.15: Cascading Stylesheets

So verwendet die Webseite aus Abbildung 4.15

- das externe Stylesheet *body1.css* wegen der Standardtextfarbe

```
body{ margin: 1.0in 1.0in; color: red}
```

- und das externe Stylesheet *body2.css* wegen des Rands und des Hintergrundbilds.

```
body { margin: 0.25in 1.0in;
        background-color: white; background-image: url(xmas.jpg);
        background-repeat: repeat-x; background-attachment: fixed }
```

Das zweite Stylesheet legt fest, dass das Bild *xmas.jpg* am oberen Rand über die Breite des Webbrowser-Fensters nebeneinander angezeigt werden soll (*repeat-x*). Zudem definiert es einen kleineren oberen Rand.

Das eingebettete Stylesheet der Webseite sieht wie folgt aus:

```
<style type="text/css">
  @import url(body1.css);
  @import url(body2.css);
  .weiss {color: white}
</style>
```

Beachten Sie, dass das Stylesheet *body2.css* nach *body1.css* importiert wird. Auf diese Weise setzt sich der kleinere obere Rand aus Stylesheet *body2.css* durch. Wenn Sie die Stylesheets in umgekehrter Reihenfolge importieren, erhalten Sie den 1-Inch-Rand.



Wenn Sie das Beispiel im Netscape Navigator nachvollziehen wollen, müssen Sie die externen Stylesheets in das eingebettete Stylesheet hineinkopieren (die import-Syntax wird vom Navigator 4 nicht unterstützt).

Der !important-Spezifizierer

Wenn Sie möchten, dass eine bestimmte Formatierung eines Stils nicht durch Stildefinitionen höherer Spezifität oder nachfolgende Stildefinitionen gleicher Spezifität übergangen wird, deklarieren Sie die Formatierung als !important.

Listing 4.9: important.html - Stildefinitionen mit Vorrang

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Vererbung</title>
  <style type="text/css">
    body    {color: white; background-color: black}
    p      {color: gray !important; font-size: 18pt}
    p.rot  {color: red; font-size: 18pt}
  </style>
</head>
<body>
<h1>Vererbung von Stilen</h1>
<p class="rot">Heraklit sah im Krieg <cite>&quot;... den Vater aller Dinge und
den König aller.&quot;
</cite></p>
</body>
</html>
```

In diesem Fall wird der Absatz mit dem Heraklit-Zitat in grauer Schrift dargestellt, obwohl die Eigenschaft `color: red` aus der `p.rot`-Stildefinition später definiert ist und die höhere Spezifität hat. Der !important-Stil setzt sich durch.

4.6 CSS-Stylesheets und die Browser

Die Formatierung mit Stylesheets ist wesentlich leistungsfähiger und fortschrittlicher als die traditionelle Formatierung mit den alten HTML-Tags und -Attributen. Dass man dennoch so viele Webseiten sieht, die auf die traditionelle Weise formatiert wurden, liegt zum einem daran, dass diese Webseiten älteren Datum sind und die Webautoren den Aufwand scheuen, den eine Aktualisierung erfordern würde, zum anderen daran, dass es immer noch viele Websurfer gibt, die Webbrowser verwenden, die Stylesheets ungenügend unterstützen.

Das Hauptproblem ist hierbei der Netscape Navigator. Laut einer Erhebung von November 2000 nutzen 35% der Websurfer den Netscape-Browser (Navigator oder Communicator). (64% verwenden den Internet Explorer und der Rest surft mit anderen Browsern.) Der Netscape Browser unterstützte die aktuellen Webstandards (neben CSS auch HTML 4 und EMCA für Web-Scripting) in den Versionen bis 4.7 jedoch nur unvollständig. Mittlerweile ist zwar der Netscape-Browser 6 verfügbar, der endlich den Anschluss gefunden hat und die Standards - insbesondere Stylesheets - besser unterstützt, doch dürfte es noch einige Zeit dauern, bis sich die Websurfer umgestellt haben.

Was ist also zu tun?

- Arbeiten Sie mit Stylesheets! Stylesheets bringen Ihnen viele Vorteile und sind die Formatierungstechnik der Gegenwart und nahen Zukunft (in der etwas ferneren Zukunft werden wir vielleicht nur noch mit XML und XSL-Stylesheets arbeiten, siehe Kapitel 21).
- Testen Sie Ihre Seiten in einer 4er-Version des Netscape Navigators. Ist das Ergebnis befriedigend, um so besser! (Wir wollen nicht zu schwarz sehen, viele Stileigenschaften werden auch vom Navigator unterstützt.) Ansonsten versuchen Sie, die gewünschte Formatierung auf einem anderen Weg zu erreichen: entweder durch andere oder zusätzliche Stildefinitionen oder indem Sie zusätzlich zu den Stylesheets auf die traditionellen HTML-Tags und -Attribute zurückgreifen - auch wenn diese deprecated sind L .

Schauen wir uns ein kleines Beispiel an.

Den meisten Ärger verursacht die Kombination von Tabellen und Stylesheets. Betrachten Sie dazu den Code aus Listing .

Listing 4.10: segenDerErde1.html - Vordergrundfarbe für Tabellen

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Segen der Erde</title>
  <meta http-equiv="Content-Style-Type" content="text/css">
  <style type="text/css">
    body { margin-top: 30px; margin-left: 30px;
      background-color: black; color: white;
      text-align: justify }
  </style>
</head>
<body>
<table border="0" width="100%" cellspacing="0" cellpadding="0" >
  <tr>
    <td width="10%">&nbsp; </td>
    <td width="450">
      <h1>Segen der Erde</h1>
      <p>Der lange, lange Pfad über
      das Moor in den Wald hinein - wer hat ihn ausgetreten? Der Mann,
      der Mensch, der erste, der hier war. Für ihn war noch kein Pfad
      vorhanden...</p>
    </td>
    <td width="10%">&nbsp; </td>
  </tr>
</table>

```

```
</body>
</html>
```

Wenn Sie diese Webseite im Netscape 6-Browser anschauen, sieht sie wie in Abbildung 4.16 aus.



Abbildung 4.16: Hier ist alles okay.

Die gleiche Webseite im Netscape Navigator 4.x ist dagegen recht enttäuschend. Eigentlich ist sie sogar absolut unbrauchbar, denn der Navigator 4 verwendet die von uns gewünschte Vordergrundfarbe (Weiß) nicht für die Tabelle. Der Websurfer sieht also schwarzen Text vor schwarzem Hintergrund. (Etwas besser ist die Darstellung im Internet Explorer 5. Dieser schluckt nur den Blocksatz in der Tabelle.)

Man kann dies auf zwei Wegen bereinigen:

- Sie definieren die Vordergrundfarbe im `<body>`-Tag mit Hilfe des deprecated-Attributs `text`:

```
<body text="white">
```

- Sie definieren die Vordergrundfarbe (und bei der Gelegenheit auch die Textausrichtung - zuliebe des Internet Explorers) in einem Stylesheet für die Tabellenzellen:

```
body { margin-top: 30px; margin-left: 30px;
        background-color: black}
td    { color: white; text-align: justify }
```

Die so formatierte Webseite sollte dann von nahezu allen Websurfern in der anvisierten Formatierung gelesen werden können.

Wo wir schon damit angefangen haben... hier eine kleine Liste von Tricks, wie man Formatierungsdefizite in den Browsern ausgleichen kann:

- Will man global die Vordergrundfarbe für eine Webseite festlegen, die eine oder mehrere Tabellen enthält, sollte man die Farbe als `color`-Stil und - aus Rücksicht auf den Navigator 4 - als `text`-Attribut des `<body>`-Tags definieren.

- Wo margin, padding und text-indent versagen, kann man Freiräume durch blinde GIFs oder leere Tabellenzellen schaffen (siehe auch Listing *Geologie.html* aus Kapitel 5.4).
- Greift eine Formatierung nicht für ein übergeordnetes HTML-Element wie <body> oder eine Tabelle oder Liste, versuchen Sie es mit den untergeordneten HTML- Elementen (Tabellenzellen, Listenelemente). Siehe obiges Beispiel.
- Greift eine Formatierung nicht für ein HTML-Element, versuchen Sie, das Element in ein formatierungsneutrales Element einzuschließen: <div>, , <a>, Tabelle ohne Rahmen und über dieses zu formatieren. So sind sich die Browser beispielsweise nicht darüber einig, was der Hintergrund einer Liste ist. Möchte man den Hintergrund für eine Liste als ein Block einfärben, empfiehlt es sich daher, die Liste in <div>-Tags oder eine einzellige Tabelle ohne Rahmen einzufassen, beispielsweise:

```

<style>
  ol { background-color: red }
  td { background-color: yellow }
  div { background-color: yellow }
</style>
...
<div>
  <ul>
    <li> This is the first item. </li>
    <li> This is the second item. </li>
    <li> This is the third item.</li>
  </ul>
</div>

```

- Wenn Sie feststellen, dass Frames nicht richtig ausgefüllt werden (das heißt, der Inhalt der eingeblendeten Webseiten reicht nicht bis zum Frame-Rand) setzen Sie die frame- Attribute marginwidth und marginheight (für Navigator 4) und leftmargin und topmargin (für IE 4) auf 0.
- Mit Stylesheets kann man Elemente auch frei auf einer Webseite positionieren (siehe Kapitel 5). Allerdings erlaubt der Navigator 4 nicht die Positionierung von HTML- Elemente, die ersetzt werden (hierzu zählen beispielsweise -Elemente, die ja letztlich nur ein Verweis auf eine Bilddatei sind und vom Browser durch den Inhalt dieser Datei ersetzt werden). Damit Bilder auch im Navigator positioniert werden, bettet man sie in <div>-Abschnitte ein und positioniert diese.

4.7 Zusammenfassung

Stylesheets sind eine moderne und leistungsfähige Technik zur Formatierung von Webseiten. Die Vorteile der Stylesheets sind:

- einheitliche Formatierung durch Stileigenschaften, die mehr oder weniger auf alle HTML-Elemente gleichermaßen anwendbar sind
- weitreichende Formatierungsmöglichkeiten durch eine Vielzahl an Stileigenschaften
- Trennung von Struktur/Inhalt und Formatierung durch eingebettete oder externe Stylesheets

Sie haben gesehen wie man

- Inline-Stile

```
<p style="color: red; text-align: right">
```

- eingebettete Stylesheets

```
<style type="text/css">
  <![CDATA[
    p {color: red; font-family: sans-serif; text-align: right}
    .unter {text-decoration: underline}
  ]]>
</style>
```

- importierte Stylesheets

```
<style>
<![CDATA[
  @import url(externessheet.css);
  #einzug { margin-left: 10%; margin-right: 10%}
]]>
</style>
```

- externe Stylesheets definiert und einbindet

```
<link rel=StyleSheet type="text/css" href="externessheet.css">
```

Und wir haben Sie darauf hingewiesen, dass man möglichst externe Stylesheets verwenden sollte (auch wenn wir dies in diesem Buch aus didaktischen Gründen - und manchmal auch aus Bequemlichkeit - selbst nicht immer tun).

Sie haben erfahren, welche Typen von Selektoren es gibt (HTML-, class-, ID- und Pseudo- Selektoren) und wie man mit Hilfe dieser Selektoren eine Verbindung zwischen den Stildefinitionen und den HTML-Elementen herstellt.

Zwischendurch haben wir uns die wichtigsten Stileigenschaften angeschaut.

Schließlich haben wir noch gesehen, wie Stile vererbt werden, wie der Browser Konflikte bei überlappenden Stildefinitionen auflöst und wie man Formatierungsprobleme aufgrund mangelnder Browserunterstützung in Angriff nehmen kann.

4.8 Fragen und Antworten

Frage:

Kann ich in meine Stylesheets Kommentare einfügen?

Antwort:

Ja, aber nicht mit der Syntax, die wir von den HTML-Kommentaren her kennen. Kommentare in Stylesheets beginnen mit der Zeichenfolge */** und enden mit **/*. Sie gleichen also den Kommentaren der Programmiersprache C.

Frage:

Welche Hintergrundfarbe hat eine Tabellenzelle?

Antwort:

Für eine Tabelle kann die Hintergrundfarbe auf 6 verschiedenen Ebenen spezifiziert werden. Der Hintergrund einer Zelle hängt davon ab, auf welcher Ebene das erste Mal ein nicht-transparenter Hintergrund definiert ist. Die einzelnen Ebenen sind: die Zelle, die Reihe, die Reihengruppe, die Spalte, die Spaltengruppe, die Tabelle.

Frage:

Ist es besser mit Inline-Stilen, eingebetteten Stylesheets oder mit externen Stylesheets zu arbeiten?

Antwort:

Inline-Stile sollte man möglichst gar nicht verwenden. Nutzen Sie stattdessen ID- oder class-Selektoren. Inline-Stile sind zwar schneller zu definieren als ID- oder class-Stildefinitionen, doch vermischt sich dadurch wieder der Code für Inhalt und Formatierung.

Gegen eingebettete Stylesheets ist prinzipiell nichts zu sagen. Externe Stylesheets haben gegenüber den eingebetteten Stylesheets jedoch den Vorteil, dass sie einfacher wiederzuverwerten und auszutauschen sind und dass sie problemlos mit XHTML-Dokumenten zusammenarbeiten.

4.9 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

Quiz

1. Wie kann man den Stylesheet-Typ angeben?
2. Warum sollte man stets den Stylesheet-Typ angeben?
3. Was ist der Unterschied zwischen padding und margin?
4. Wie kann man Text unterstreichen ?
5. Zur Erinnerung an Kapitel 3: Wie rückt man den Inhalt von Tabellenzellen von den Zellenrändern ab?
6. Was bedeutet Vererbung im Sinne der CSS-Spezifikation?
7. Welches Stylesheet hat die höhere Priorität (unter der Annahme, dass alle aufgeführten Stildefinitionen auf ein gegebenes Element anwendbar sind)?
 - ul
 - li.meineinzug
 - ul ul li

Übungen

1. Webs aus mehreren Webseiten basieren üblicherweise auf einem Grund-Design, das für alle Webseiten gleich ist. Zu einem solchen Grund-Design könnten zum Beispiel aufeinander abgestimmte Vorder- und Hintergrundfarbe (oder Hintergrundbild), Hyperlinkfarben, Listensymbole und Schriften für Überschriften und Textabsätze gehören. Entwerfen Sie ein solches Design als externes Stylesheet.
2. Setzen Sie eine Webseite mit einer Überschrift, einem Textabsatz, einigen Hyperlinks und einer Liste auf. Weisen Sie dann der Webseite Ihr Design zu und beobachten Sie, wie schnell man mit externen Stylesheets formatieren kann.

Tag 5

Positionieren mit Stylesheets

Die Verwendung von Stylesheets bricht mit einem der Grundprinzipien des World Wide Webs und speziell des Designs von Webseiten: der Autor der Webseite gibt die Formatierung dezidiert vor, statt Empfehlungen auszusprechen, die clientseitig vom Webbrowser umgesetzt werden. Dies betrifft nicht nur das Erscheinungsbild der einzelnen Elemente (durch Festlegung einer Schriftart, einer Vorder- und Hintergrundfarbe, etc.), sondern auch die Positionierung der Elemente auf der Webseite.

Die Themen heute:

- Vergleich zwischen traditioneller und CSS-Positionierung
- Drei Formen der CSS-Positionierung
- Überlagerung von HTML-Elementen
- CSS-Positionierung versus Tabellen
- Clipping
- HTML-Elemente ausblenden
- Textfluss um Bilder

5.1 Vor- und Nachteile der traditionellen Positionierung

Die Grundidee des Webdesigns ist es, die verschiedenen Elemente (Textblöcke, Bilder, Linien, Applets, etc.) relativ zueinander zu positionieren und es dem Webbrowser zu überlassen, diese relative Anordnung soweit es geht sicherzustellen.

Vergegenwärtigt man sich, dass Sie als Webautor bei der Erstellung einer Webseite weder wissen, welchen Browser Ihre Leser verwenden, noch welche Layout-Techniken die Browser Ihrer Leser unterstützen (Grafik, Skriptsprachen, Java, CSS), noch wie groß der Anzeigebereich des Browsers ist, so ist der Entschluss, die endgültige Formatierung und das endgültige Layout dem jeweiligen Browser zu überlassen, leicht nachzuvollziehen.

Schauen Sie sich zum Beispiel die Seite aus Abbildung 5.1 an.

Die Webseite enthält einfach einen Textblock, gefolgt von einer Grafik. An sich nichts Aufregendes - ein normales Textdokument könnte genauso aussehen.

Interessant wird es aber, wenn man sich den HTML-Text dieser Seite anschaut. Im einfachsten Fall wird die Webseite im Editor in genau der gleichen Weise aufgesetzt, wie ein Textdokument. Sie nutzen dann automatisch die relative Positionierung durch den Browser. Sie hätten aber auch die Möglichkeit, die Position der eingebetteten Grafik durch ein Stylesheet absolut festzulegen. Sie brauchen dazu nur für das `img`-Tag die Stileigenschaft `position` auf `absolute` zu setzen und die `x,y`-Koordinate der linken oberen Bildecke anzugeben (Stileigenschaften `left` und `top`).



Abbildung 5.1: Grafik unter einem Textblock

Relative Positionierung	Absolute Positionierung
<pre> <!DOCTYPE ... <html> <head> <title>Traditionelle Positionierung</title> <style type="text/css"> body { background-color: #cdcdcd } </style> </head> <body> <h1>Flekkefjord</h1> <p>Flekkefjord ist eine kleine Hafenstadt an der Südküste Norwegens, ungefähr in der Mitte zwischen Kristiansand und Stavanger. Trotz des unermüdlichen Stroms an Pendlern und Touristen, die auf dem Weg </pre>	<pre> <!DOCTYPE ... <html> <head> <title>Absolute Positionierung</title> <style type="text/css"> body { background-color: #cdcdcd } img {position: absolute; top: 150; left : 10} </style> </head> <body> <h1>Flekkefjord</h1> <p>Flekkefjord ist eine kleine Hafenstadt an der Südküste Norwegens, ungefähr in der Mitte zwischen </pre>

von Kristiansand nach Stavanger durch das Städtchen ziehen, hat sich Flekkefjord den Charakter des verschlafenen Hafenstädtchens bewahrt.</p>

</body>

</html>

Kristiansand und Stavanger. Trotz des unermüdlichen Stroms an Pendlern und Touristen, die auf dem Weg von Kristiansand nach Stavanger durch das Städtchen ziehen, hat sich Flekkefjord den Charakter des verschlafenen Hafenstädtchens bewahrt.</p>

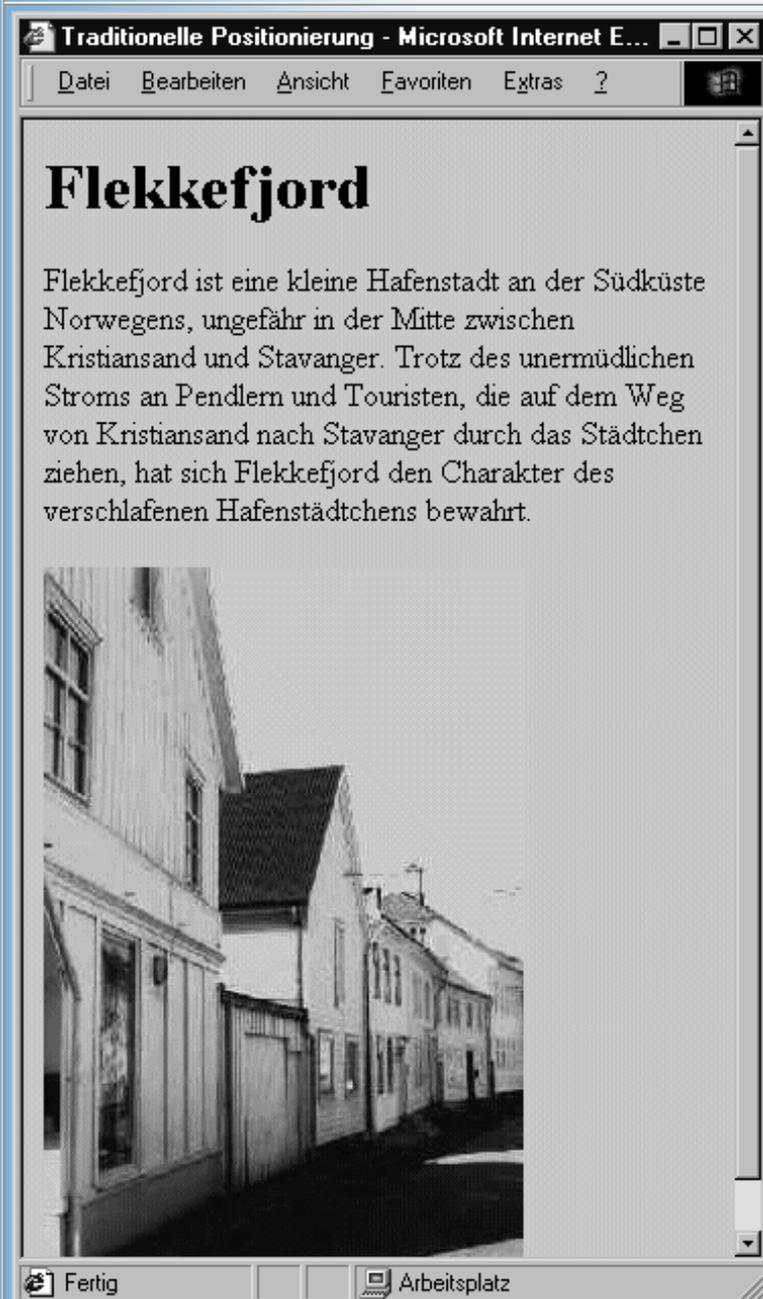
</body>

</html>

Tabelle 5.1: Traditionelle relative und absolute Positionierung

Solange Sie die Abmaße des Browsers (oder der Editorvorschau) nicht ändern, ist alles in Ordnung. Der Unterschied zwischen relativer und absoluter Positionierung macht sich aber bemerkbar, sowie Sie das Browser-Fenster verändern, es beispielsweise verschmälern.

Traditionelle Positionierung



Absolute Positionierung

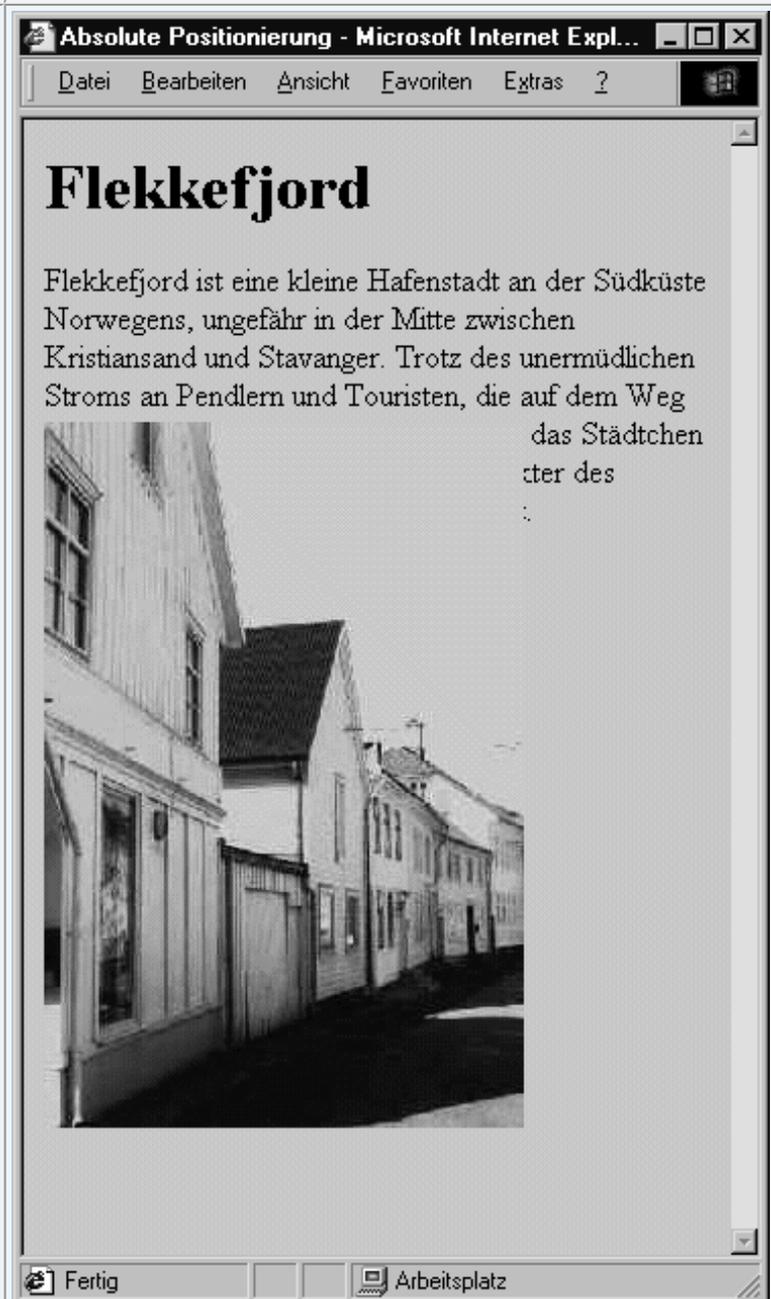


Tabelle 5.2: Relative und absolute Positionierung

Wird das Browser-Fenster verschmälert, bricht der Browser den Text um, damit dieser ohne horizontales Scrollen gelesen werden kann. Bei der relativen Positionierung rückt der Browser automatisch die Grafik weiter nach unten, so dass sie nach wie vor unter dem Textblock angezeigt wird. Wurde die Position der Grafik dagegen absolut angegeben, bleibt Sie an dieser Position - Text und Grafik überlagern sich. Man kann dies korrigieren, indem man den Textblock in einen <div>-Block einfasst und diesen ebenfalls absolut positioniert oder indem man unter Zuhilfenahme einer Skriptsprache das -Element dynamisch positioniert.



Absolute Positionierung einzelner Elemente kann zu unerwünschten Überlagerungen führen

Vorteile der absoluten Positionierung

Die absolute Positionierung kann aber auch sehr hilfreich sein - beispielsweise

- wenn man gerade an der Überlagerung von HTML-Elementen interessiert ist oder
- wenn man die Anpassung des Seitenlayouts an die Breite des Browser-Fensters unterbinden will
- wenn man eine Alternative zu Tabellen oder 1-Pixel-Grafiken, die auch häufig zur Positionierung verwendet werden, sucht.

5.2 Grundlagen der CSS-Positionierung

Bevor wir uns die einzelnen Attribute für die CSS-Positionierung anschauen, sollten wir uns mit den Grundideen der Positionierung von HTML-Elementen vertraut machen.

HTML-Positionierung

Der grundlegende Algorithmus zur Positionierung von HTML-Elemente ist an sich ganz einfach:

- Blockelemente (<h1>, <p>, <div> etc.) werden in der Reihenfolge ihres Auftretens untereinander positioniert.
- Inline-Elemente (, , etc.) werden in der Reihenfolge ihres Auftretens nebeneinander positioniert. Passen Sie nicht in eine Reihe kann der Browser sie umbrechen.

So wird beispielsweise aus dem folgenden HTML-Code:

```
...  
<body>  
<h1>HTML-Positionierung</h1>  
<p>Erster Absatz mit Bild :  und Textpassage:  
<em>Dies ist ein eingebettetes HTML-Element</em> </p>  
<p>Zweiter Absatz </p>  
</body>  
...
```

die Webseite aus Abbildung 5.2.



Abbildung 5.2: Positionierung nach Reihenfolge des Auftretens

HTML kennt nur wenige und zudem recht eingeschränkte Möglichkeiten, auf diese Form der Positionierung Einfluss zu nehmen: die Verwendung von Tabellen (siehe Kapitel 3.1) und das Herausreißen von Bildern aus dem Fluss der umliegenden HTML-Elemente (siehe Abschnitt 5.6).

Mit CSS-Positioning ist es dagegen möglich, zwischen vier Positionierungsschemas zu wählen. Sie müssen dazu nur der Stileigenschaft `position` einen der folgenden Werte zuweisen:

- `static`
- `relative`
- `absolute`
- `fixed`

Beginnen wir mit dem Schema, das Sie ohne Zweifel am meisten interessieren wird: der absoluten Positionierung.

Absolute Positionierung

Absolute Positionierung bedeutet, dass Sie die Koordinaten, an denen das Element eingeblendet werden soll, explizit angeben. Wenn Sie ein Element, beispielsweise ein Bild absolut positionieren wollen, gehen Sie wie folgt vor:

1. Setzen Sie die `position`-Eigenschaft für das Element auf `absolute` (sie können dies über ein Stylesheet oder das `style`-Attribut tun).

```
<img style="position: absolute"
```

2. Dann positionieren Sie das Element, indem Sie den `top`- und `left`-Eigenschaften Werte zuweisen.

```
<img style="position: absolute; top: 20px; left: 10px" src=... />
```



Der Netscape Navigator 4 erlaubt keine CSS-Positionierung ersetzter Elemente (im Falle des -Tags wird das Element durch das Bild »ersetzt«, auf dessen Datei sein src-Attribut verweist).

Die Eigenschaften left und top

Wie genau erfolgt die Positionierung über left und top?

Erinnern Sie sich an das Box-Modell, das wir in Kapitel 4.3.2 besprochen haben? Jedes HTML-Element liegt aus Sicht des Browsers in einer unsichtbaren rechteckigen Box, die seinen Inhalt - inklusive Füllraum und Rand - einschließt. Wenn der Browser die verschiedenen HTML-Elemente auf der Webseite positioniert, positioniert er keine Textabsätze oder Bilder, sondern Boxen (mit verschiedenen Inhalten). Was dem Browser recht ist, kann uns nur billig sein - sprich, die Eigenschaften left und top beziehen sich auf die Box um das zu positionierende HTML-Element.

- left - Die Eigenschaft left legt fest, wie weit der linke Rand der Box des Elements vom linken Rand des Positionierungskontextes nach rechts abgerückt werden soll.
- top - Die Eigenschaft top legt fest, wie weit der obere Rand der Box des Elements vom oberen Rand des Positionierungskontextes nach unten abgerückt werden soll.



Neben left und top kann man noch die Stileigenschaften bottom und right zur Positionierung nutzen.

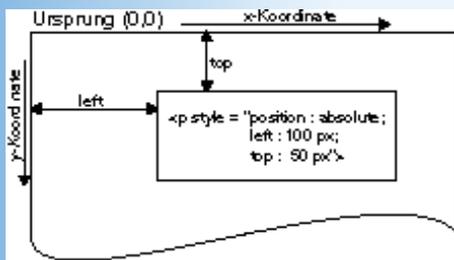


Abbildung 5.3: Positionierung mit left und top

Jetzt müssen wir nur noch klären, was ein Positionierungskontext ist.



Der Positionierungskontext ist das Koordinatensystem, in dem HTML-Elemente durch explizite Positionsangaben (left, top = Position der linken oberen Ecke der Box des HTML-Elements) oder durch ihre Einbettung im Fluss des HTML-Codes ausgerichtet werden.

Jede Webseite verfügt über einen Standardpositionierungskontext, der an das <html>-Tag gebunden ist. Seine Koordinaten und Abmaße werden vom Browser definiert und können nicht verändert werden. Sofern man von der CSS-Positionierung keinen Gebrauch macht, gibt es nur diesen einen Positionierungskontext. Wird aber für ein HTML-Element die Stileigenschaft position auf absolute oder relative gesetzt, bildet dieses Element einen eigenen Positionierungskontext, der für alle eingebetteten Elemente (soweit vorhanden) gilt.

Betrachten Sie dazu das folgende Beispiel, in dem die Textpassage (-Abschnitt) mit der Stil-ID aussen absolut positioniert wird. Der in dieser Textpassage eingebettete -Abschnitt mit der ID innen wird ebenfalls absolut positioniert - und zwar mit genau den gleichen Werten für top und left. Trotzdem überlagern sich die beiden absolut positionierten Elemente nicht, denn der äußere Abschnitt wird im Positionierungskontext der HTML-Seite

positioniert, während der innere Abschnitt im Positionierungskontext des umliegenden, absolut positionierten Abschnitts verschoben wird.

HTML-Code	Ansicht im Browser
<pre> ... <style type="text/css"> #aussen {position: absolute; top: 2cm; left: 3cm; width: 5cm; color: red} #innen {position: absolute; top: 2cm; left: 3cm; width: 5cm; font-weight: bold} </style> ... <body> <p>o Beginn der äußeren Textpassage. Innerer Text. Ende der äußeren Textpassage. o </p> </body> </html> </pre>	

Tabelle 5.3: Absolute Positionierung



Wenn Sie Elemente absolut positionieren, achten Sie auf die verwendete Maßeinheit. Absolute Maßeinheiten wie cm, in oder pt sind nicht auf allen Systemen gleich. Konsistentere Ergebnisse erzielt man meist mit em, px oder %.

Überlagerung von Elementen

Wenn Sie HTML-Elemente absolut positionieren, kann es leicht passieren, dass sich die positionierten Elemente überlagern (unter Umständen ist dies überhaupt der Grund, warum Sie die Elemente absolut positionieren). Wenn sich zwei Elemente überlagern, stellt sich natürlich sofort die Frage, welches Element oben und welches unten liegt.

- Absolut positionierte Elemente liegen immer über den nicht positionierten Elementen.
- Wurden für zwei positionierte Elemente z-index-Werte angegeben, liegt das Element mit dem höheren z-index-Wert über dem Element mit dem niedrigeren Wert (der Vorgabewert ist 0).
- Zwei absolut positionierte Elemente mit gleichem z-index-Wert werden in der Reihenfolge überlagert, in der sie positioniert werden.



Wenn Sie Bilder mit grafischen Objekten überlagern, verwenden Sie GIF- oder PNG-Bilder mit transparentem Hintergrund.

Zusammenfassung

Absolut positionierte Elemente haben vier wichtige Eigenschaften:

- Ihre Positionierung erfolgt gemäß ihrer top- und left-Werte im umliegenden Positionierungskontext.
- Jedes absolut positionierte Element erzeugt einen eigenen Positionierungskontext für HTML-Elemente, die in ihm eingebettet sind.
- Absolut positionierte Elemente werden gemäß ihres z-Index übereinander gestapelt. Ein höherer z-Index bedeutet, dass das Element über andere Elemente angezeigt wird.
- Absolut positionierte Elemente werden aus dem »normalen« Aufbau ganz herausgenommen (das heißt, im Gegensatz zur relativen Positionierung hinterlassen sie keine »Löcher« an ihrer ursprünglichen Position, siehe unten).

Relative Positionierung

Die relative Positionierung (position: relative) kommt einer Verschiebung am nächsten. Man stellt sich das am besten so vor, dass die Webseite zuerst so angezeigt wird, als ob das Element überhaupt nicht positioniert würde. Dann wird das Element aus der formatierten Seite ausgeschnitten und von seiner Anfangsposition um die top- und left-Werte verschoben.

Relativ positionierte Elemente haben vier wichtige Eigenschaften:

- Ihre Positionierung erfolgt - gemäß ihrer top- und left-Werte - relativ zu ihrer »ursprünglichen« Position.
- Jedes relativ positionierte Element erzeugt einen eigenen Positionierungskontext für HTML-Elemente, die in ihm eingebettet sind.
- Relativ positionierte Elemente werden gemäß ihres z-Index übereinander gestapelt. Ein höherer z-Index bedeutet, dass das Element über andere Elemente angezeigt wird.
- Relativ positionierte Elemente hinterlassen an ihrer ursprünglichen Position einen Leerraum.

HTML-Code	Ansicht im Browser
<pre><style type="text/css"> #aussen {position: relative; top: 3cm; left: 3cm; width: 5cm; color: red} #innen {font-weight: bold} </style></pre>	

```

...
<body>
<p>oooooooooooooooooooooooooooooooooooo
<span id="ausсен">Beginn der äusseren Textpassage.
<span id="innen">Innerer Text.</span> Ende der
äusseren Textpassage. </span>oooooooooooo
oooooooooooooooooooo </p>
</body>
</html>

```

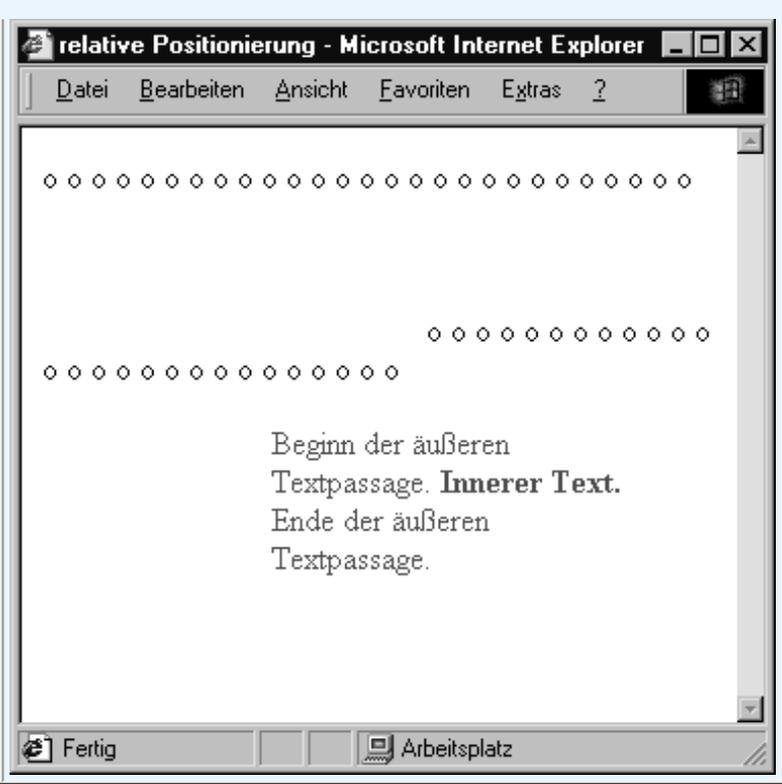


Tabelle 5.4: Relative Positionierung

Statische und fixierte Positionierung

Statische Positionierung (`position: static`) bedeutet, dass das HTML- Positionierungsschema verwendet wird. (Positionierung durch den Browser nach dem Ort der Definition; Werte für die Attribute `left` und `top` werden ignoriert.)

Fixierte Positionierung (`position: fixed`) bedeutet, dass ein Element absolut positioniert wird und im Browser nicht zusammen mit dem Webinhalt gescrollt wird (beim Ausdruck erscheint das Element an der entsprechenden Position auf jeder Seite). Die fixierte Positionierung wird derzeit noch von keinem Browser unterstützt.

5.3 Die Positionierungsattribute

In Tabelle 5.5 sind die verschiedenen Attribute aufgeführt, die Ihnen für die CSS- Positionierung zur Verfügung stehen.

Attribut	Werte	Beschreibung
position	static	erlaubt keine explizite Positionierung
	absolute	erlaubt explizite Positionierung relativ zum übergeordneten Positionierungskontext und erzeugt einen eigenen Positionierungskontext für eingebettete Elemente.
	relative	erlaubt explizite Positionierung relativ zur ursprünglichen Lage und erzeugt einen eigenen Positionierungskontext für eingebettete Elemente.
left	auto	Abstand des linken Randes der Box vom linken Rand des Positionierungskontextes. Der Ursprung der Verschiebung hängt von dem Wert von position ab.
	Länge (em, px, cm, in...)	
	Prozent (%)	

top	auto Länge (em, px, cm, in...) Prozent (%)	Abstand des oberen Randes der Box vom oberen Rand des Positionierungskontextes. Der Ursprung der Verschiebung hängt von dem Wert von position ab.
bottom	auto Länge (em, px, cm, in...) Prozent (%)	Abstand des unteren Randes der Box vom unteren Rand des Positionierungskontextes. Der Ursprung der Verschiebung hängt von dem Wert von position ab.
right	auto Länge (em, px, cm, in...) Prozent (%)	Abstand des rechten Randes der Box vom rechten Rand des Positionierungskontextes. Der Ursprung der Verschiebung hängt von dem Wert von position ab.
z-index	auto Wert: 1, 2, 3...	Elemente mit höheren Werten werden bei Überlagerung über Elemente mit niedrigeren Werten gestapelt. Elemente mit gleichem z-Index werden gemäß ihrer Position im HTML-Code von unten nach oben gestapelt.

Tabelle 5.5: Attribute für die CSS-Positionierung

5.4 Freies Webdesign dank CSS-Positionierung

Webseiten, die als Eintrittsseiten in größere Webs fungieren, sind oftmals so aufgebaut, dass sie zum einem das Thema der Website transportieren, zum anderen Links zu den Webseiten der nächsten Ebene bereitstellen.

Die Seite aus Abbildung 5.4 ist hierfür ein Beispiel. Sie lädt Sie ein, Lateinamerika zu besuchen, und bietet Ihnen verschiedene grafische Links an, über die Sie sich über Land und Leute informieren können.



Abbildung 5.4: Webdesign mit Hilfe der CSS-Positionierung

Der Quelltext dieser Seite sehen Sie in Listing 5.1.

Listing 5.1: freiesDesign.html

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Freies Design</title>
  <style type="text/css">
    body { background-image: url(blaustreifen.jpg);
          background-repeat: repeat-y;
          background-color: white }
    div { position : absolute }
    #latin      { margin-left: 110px}
    #geschichte { top: 230px; left: 120px}
    #leute      { top: 330px; left: 60px}
    #land       { top: 330px; left: 220px}
    #kultur     { top: 150px; left: 576px}
    #wirtschaft { top: 45px; left: 512px}
  </style>
</head>
<body>
<h2>Willkommen in Lateinamerika</h2>

<div id="geschichte">
  
</div>
<div id="leute">
   </p>
</div>
<div id="land">
  
</div>
<div id="kultur">
  
</div>
<div id="wirtschaft">
  
</div>
</body>
</html>

```

Wenn Sie sich den HTML-Code anschauen, werden Sie feststellen, dass alle absolut zu positionierenden ``-Elemente in `<div>`-Elemente eingefasst wurden und dass es diese sind, die in Wirklichkeit absolut positioniert werden. Dies liegt daran, dass der Netscape Navigator 4 im Gegensatz zum Internet Explorer oder dem Netscape 6-Browser keine absolute Positionierung zu ersetzender Elemente (wie z.B. ``-Elemente) erlaubt. Um auch den Netscape Navigator 4 zu unterstützen, wurden die ``-Elemente daher in formatierungsneutrale `<div>`-Elemente eingeschlossen und mit diesen zusammen positioniert.

Der blaue Randstreifen ist ein Hintergrundbild, das entlang der y-Achse wiederholt wird:

```

body { background-image: url(blaustreifen.jpg);
      background-repeat: repeat-y;
      background-color: white }

```



Wenn Sie ein Hintergrundbild wählen, das nicht das gesamte Browserfenster ausfüllt (wie in obigem Beispiel), sollte man unbedingt die gewünschte Hintergrundfarbe explizit setzen. Verlassen Sie sich keinesfalls darauf, dass die Websurfer Ihre Webbrowser so eingestellt haben, dass der Hintergrund immer Weiß ist.

Die Karte von Lateinamerika, die wie ein Hintergrundbild aussieht, ist in Wirklichkeit ein normal eingebettetes Bild, das um 110 Pixel nach rechts verschoben ist.

```

#latin { margin-left: 110px}

```

Das Einzige, was der Webseite noch fehlt, sind die Hyperlinks zu den positionierten Bildern.

CSS-Positionierung versus Tabellen

Bevor man die Möglichkeit der CSS-Positionierung in den großen Browsern nutzen konnte, wurden Layouts wie für die Lateinamerika-Seite aus dem vorangehenden Abschnitt meist mit Hilfe von Tabellen aufgebaut. Es stellt sich daher die Frage, ob uns die CSS-Positionierung hier überhaupt Vorteile bringt oder ob nicht doch die Verwendung von Tabellen einfacher wäre.

Vorteile der Tabellen

- Viele HTML-Editoren unterstützen die Arbeit mit Tabellen durch spezielle Befehle und WYSIWYG-Darstellung. Dies spart dem Webdesigner Tipparbeit und erlaubt ihm, den Aufbau der Tabelle während der Bearbeitung zu kontrollieren.
- Mit Tabellen kann man eine Webseite in einzelne Abschnitte aufteilen
- Tabellen eignen sich meist besser zur Seitengestaltung, wenn die Positionierung der Elemente in gewissen Maßen an die Breite des Browserfenster angepasst werden soll.

Vorteile der CSS-Positionierung

- Die CSS-Positionierung ermöglicht eine echte absolute Positionierung (vollkommen unabhängig von Größe des Browser-Fensters).
- Mittels CSS-Positionierung lassen sich am besten einzelne Elemente frei positionieren
- Die CSS-Positionierung erlaubt die Überlagerung von Elementen.

Der letzte Punkt wurde im Lateinamerika-Beispiel gleich zweimal ausgenutzt: einmal bei der Überlagerung der Bilder für Industrie und Kultur, zum anderen bei der Überlagerung der Landkarte. In einer Tabelle müssten Sie dazu die Landkarte als Hintergrundbild definieren. Hat man aber bereits wie in obigem Beispiel bereits ein anders Bild als Hintergrundbild ausgesucht, bleibt nur der Ausweg, die einzelnen Bilder zu einem großen Bild zu verschmelzen und dieses wieder in Teilbilder aufzubrechen, die man auf die Tabellenzellen verteilt (siehe auch Übung 3.2).

Tabellen und CSS-Positionierung gemeinsam nutzen

Schließlich besteht die Möglichkeit, Elemente innerhalb einer Tabelle absolut zu positionieren. Dabei sollten Sie folgende Empfehlungen beachten:

- Legen Sie die Tabellenbreite absolut fest (und nicht in Prozent der Seite), um Verschiebungen und Überlagerungen durch variable Browser-Fenstergrößen zu unterbinden.
- Deklarieren Sie Tabellenzellen, in denen Sie Elemente positionieren wollen, ohne Angabe von top- und left-Werten als absolut positioniert (`td { position: absolute }`). Sie erreichen damit, dass die Zelle den Positionierungskontext für die in ihr enthaltenen Elemente bildet und letztere folglich relativ zu der linken oberen Ecke der Zelle positioniert werden.

Aber nicht immer ist CSS-Positionierung die elegantere Lösung. Betrachten Sie zum Beispiel die Webseite aus Abbildung 5.5.



Abbildung 5.5: Aufwendig positionierte Webseite

Ein Design wie in Abbildung 5.5 baut man am Besten mit Hilfe einer Tabelle auf, siehe Abbildung 5.6.

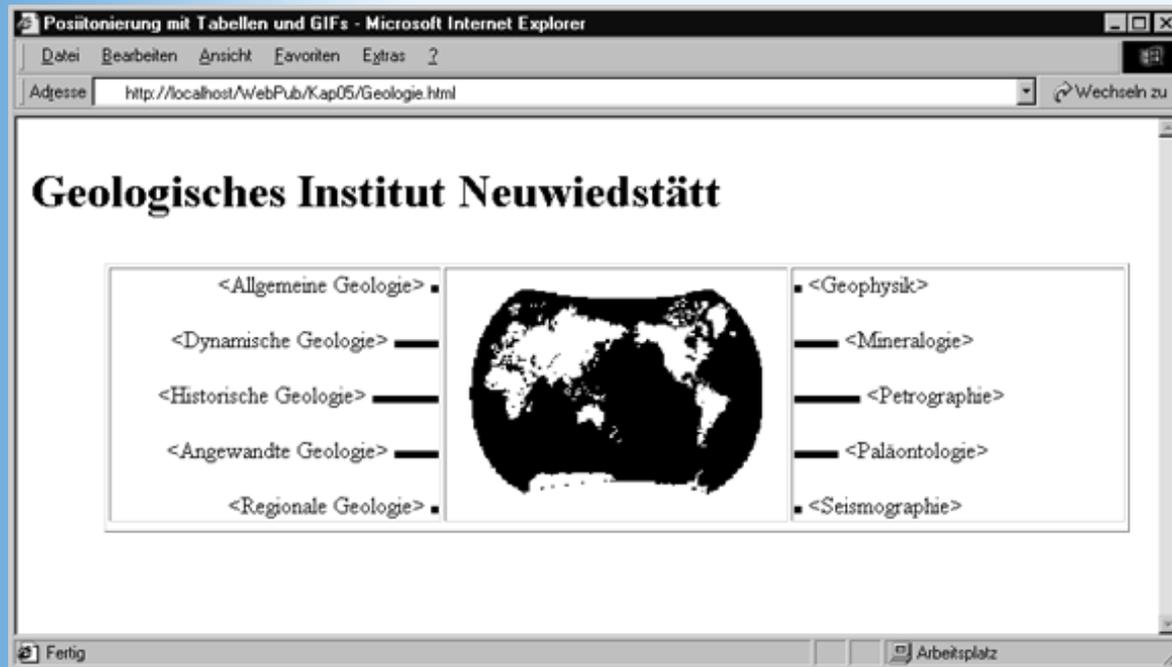


Abbildung 5.6: Positionierung mit Tabellen und blinden GIFs

Die bogenförmige Ausrichtung der Hyperlinks (im Beispiel nur durch Text angedeutet) erreicht man am einfachsten durch die Einschaltung von blinden GIFs (hier *spacer.gif*).

spacer.gif ist ein einfaches GIF-Bild von genau einem Pixel in transparenter Farbe. Indem man dieses Bild den Hyperlink-Texten in der linken Zelle nachstellt (bzw. den Hyperlink-Texten in der rechten Zelle voranstellt) und dann die Weiten der GIFs für die untereinander liegenden Hyperlinks in gleichmäßigen Schritten erhöht und wieder erniedrigt, kann man ohne große Mühe einen perfekten Bogen erzeugen.

Als HTML-Code sieht das dann wie folgt aus.

Listing 5.2: Geologie.html - Positionierung mit Tabellen und GIFs

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Posiitonierung mit Tabellen und GIFs</title>
  <style type="text/css">
    body { margin-top: 2em; background-color: maroon; color: white}
    h1   { margin-bottom: 1em }
    div  { margin-left: 50px }
  </style>
</head>
<body text="white">
<h1>Geologisches Institut Neuwiedstätt</h1>
<div>
<table border="0" width="700">
  <tr>
    <td width="33%" align="right">
      <p>&lt;Allgemeine Geologie&gt;
        
      </p>
      <p>&lt;Dynamische Geologie&gt;
        
```

```

</p>
<p>&lt;Historische Geologie&gt;
  
</p>
<p>&lt;Angewandte Geologie&gt;
  
</p>
<p>&lt;Regionale Geologie&gt;
  
</p>
</td>
<td valign="middle" align="center">
  
</td>
<td width="33%">
  <p>
    &lt;Geophysik&gt;
  </p>
  <p>
    &lt;Mineralogie&gt;
  </p>
  <p>
    &lt;Petrographie&gt;
  </p>
  <p>
    &lt;Paläontologie&gt;
  </p>
  <p>
    &lt;Seismographie&gt;
  </p>
</td>
</tr>
</table>
</div>
</body>
</html>

```

Warum Tabellen und blinde GIFs?

Nun, die Positionierung mit der Tabelle geht in diesem Fall natürlich viel schneller als die explizite CSS-Positionierung und ist zudem wesentlich wartungsfreundlicher. Statt der blinden GIFs hätte man auch auf die Stileigenschaft padding zurückgreifen können, doch wird diese von Navigator 4 nicht in Tabellenzellen unterstützt. Daher die etwas altmodische Einrückung mittels eines blinden GIFs.



Wenn Sie blinde GIFs verwenden, achten Sie darauf, dass diese transparent sind!

5.5 Sichtbarkeit und Clipping

Die CSS2-Spezifikation führte nicht nur das Positionieren mit Stylesheets ein, sie brachte auch Unterstützung für das Clipping und Ausblenden von HTML-Elementen. Von Clipping spricht man, wenn nur ein Teil eines HTML-Elements sichtbar bleibt.

Clipping

Mit Hilfe der Stileigenschaft `overflow` kann man den sichtbaren Bereich eines HTML- Elements beschneiden (und gegebenenfalls Bildlaufleisten zum Scrollen des sichtbaren Bereichs einrichten).

Zur Beschränkung des Anzeigebereichs reduziert man die Box-Abmaße. Mit Hilfe der Stileigenschaften `width` und `height` kann man für Blockelemente (und explizit positionierte Inline-Elemente) die Box-Abmaße festlegen:

```
style="width: 450px; height: 200px"
```

Dies allein führt aber noch nicht zu dem gewünschten Ergebnis, da der Inhalt eines HTML-Elements über die Box hinaus reichen kann. Will man erreichen, dass der Boxinhalt an den Grenzen der Box abgeschnitten wird, muss man noch die Eigenschaft `overflow` auf `hidden` setzen:

```
style="width: 450px; height: 200px; overflow: hidden"
```

Will man den Anzeigebereich mit Bildlaufleisten ausstatten, verwendet man `scroll` statt `hidden`.

```
style="width: 450px; height: 200px; overflow: scroll"
```

Will man selbst explizit vorgeben, welcher Ausschnitt des HTML-Elements sichtbar sein soll, legt man den Ausschnitt über die Stileigenschaft `clip` fest:

```
style="width: 450px; height: 200px; overflow: hidden;
      clip: rect(50px 300px 100px 50px)"
```



Die Koordinaten für das Clipping-Rechteck werden in der Reihenfolge oben, rechts, unten, links angegeben.

Listing 5.3: clipping.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Overflow</title>
</head>
<body>
<h1>Gedichtesammlung</h1>
<div style="margin-left: 50px">
<table border="0" width="700" cellspacing="10px" style="table-layout: fixed">
<colgroup span="2">
  <col width="150">
  <col width="500">
</colgroup>
<tr valign="top">
  <td width="150">
    <h3>Abschied</h3>
  </td>
  <td width="500">
    <div style="padding: 2em; width: 450px; height: 200px;
```

```

        overflow: scroll;
        border-style:solid; border-width: 3px">
<p>Steig doch vom Pferd, komm, laß uns einen Becher leeren,<br />
    Verrate mir, wohin die Reise geht!<br />
    ...</p>
<p>Wang Wei</p>
</div>
</td>
</tr>
<tr valign="top">
    <td width="150">
        <h3>Der Panther</h3>
    </td>
    <td width="500">
        <div style="padding: 2em; width: 450px; height: 200px;
            overflow: scroll;
            border-style:solid; border-width: 3px">
<p>Sein Blick ist vom Vorübergehn der Stäbe<br />
    so müd geworden, daß er nichts mehr hält.<br />
    Ihm ist, als ob es tausend Stäbe gäbe<br />
    ...</p>
<p>Rilke</p>
</div>
    </td>
</tr>
</table>
</div>
</body>
</html>

```

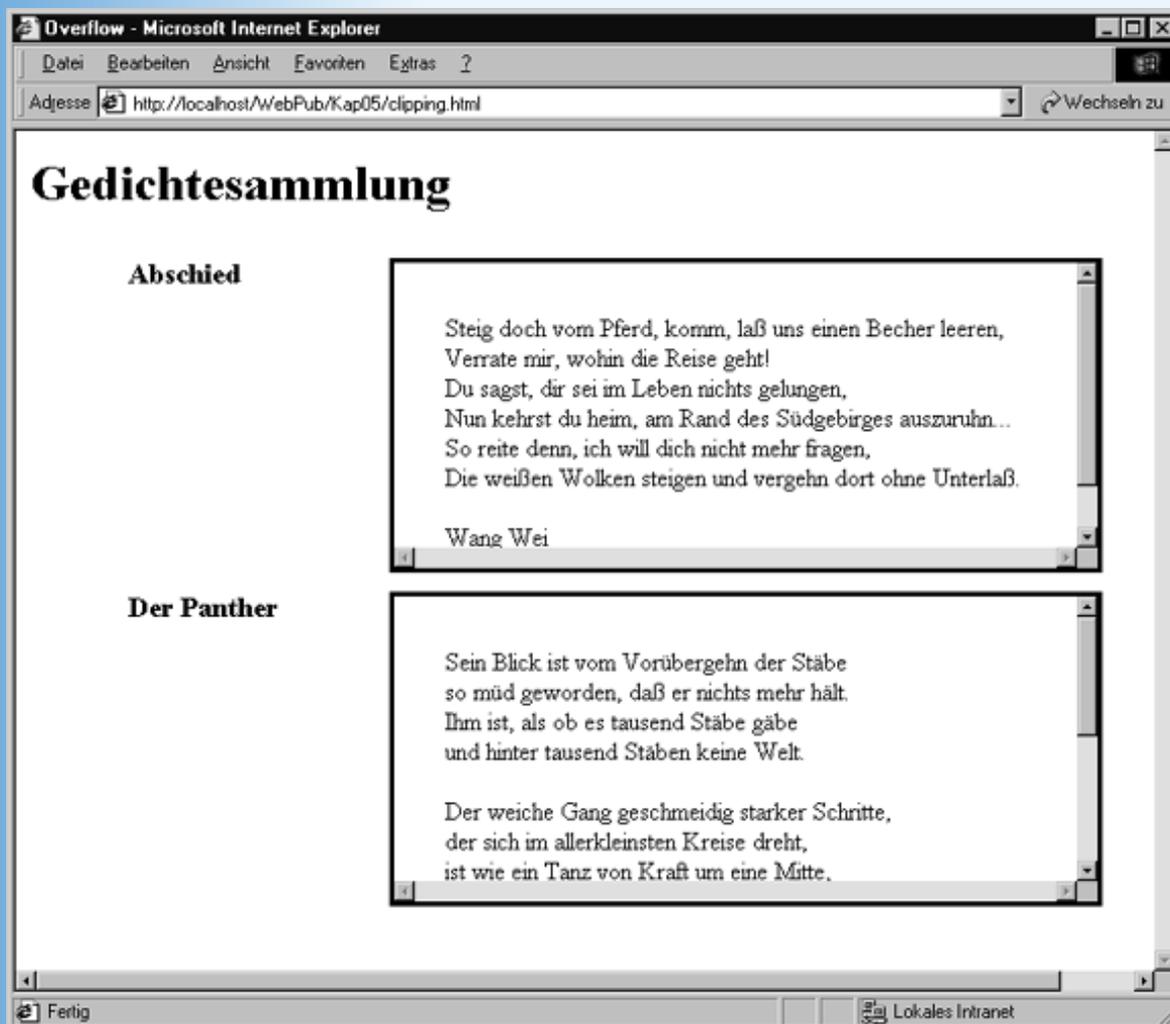


Abbildung 5.7: Design mit Clipping-Bereichen



Das Design aus Abbildung 5.7 wird sowohl im Internet Explorer als auch im Netscape 6-Browser korrekt angezeigt. Der Navigator 4 unterstützt das Clipping nicht wie gewünscht. Er zeigt den Inhalt des geclippten Elements vollständig an, so dass wenigstens keine Information verloren geht.

Sichtbarkeit

Schließlich gibt es die Möglichkeit, ein Element durch Setzen der Stileigenschaft `visibility` gänzlich ein- und auszublenden. Sie brauchen die Eigenschaft dazu nur auf `visible` beziehungsweise `hidden` zu setzen.

Diese Eigenschaft hat nicht nur für die dynamische Gestaltung mittels JavaScript ein enormes Potential, sie kann auch bei der Einrichtung suchmaschinenfreundlicher Webseiten helfen. Viele Suchmaschinen kontrollieren nämlich, ob die im `<head>`-Abschnitt aufgelisteten Stichwörter auch wirklich im Text der Webseite vorkommen. Handelt es sich aber um eine rein grafische Eingangsseite, verfügt diese unter Umständen über gar keinen Text, weil dieser das Design stören würde. In solchen Fällen kann man sich damit behelfen, dass man am Ende der Webseite einen passenden Text, in dem die betreffenden Stichwörter alle verwendet werden, als `<p>`-Absatz aufsetzt und dieses Absatz dann mittels `<p style="visibility: hidden">` unsichtbar werden lässt.



Die Eigenschaft `visibility` wird von älteren Browsern oder auch dem Navigator 4 nicht voll unterstützt. Gliedern Sie den Text also möglichst so in die Eingangsseite ein, dass er nicht zu sehr stört, sollte er doch angezeigt werden.

5.6 Textfluss um Bilder

Kommen wir noch einmal auf die Positionierung von HTML-Elementen zurück. Alte HTML-Hasen kennen noch die traditionelle Ausrichtung von Bildern mit Hilfe des `align`-Attributs.

- Die Werte `bottom`, `middle` und `top` dienen¹ dazu, ein ``-Bild, das in einen Textabsatz eingebettet ist, an der Grundlinie der umgebenden Textzeile auszurichten.
- Die Werte `left` und `right` dienen dazu, ein ``-Bild aus dem Textfluss zu lösen und an den linken oder rechten Rand zu schieben. Der nachfolgende Absatz wird danach rechts beziehungsweise links neben dem Bild dargestellt.

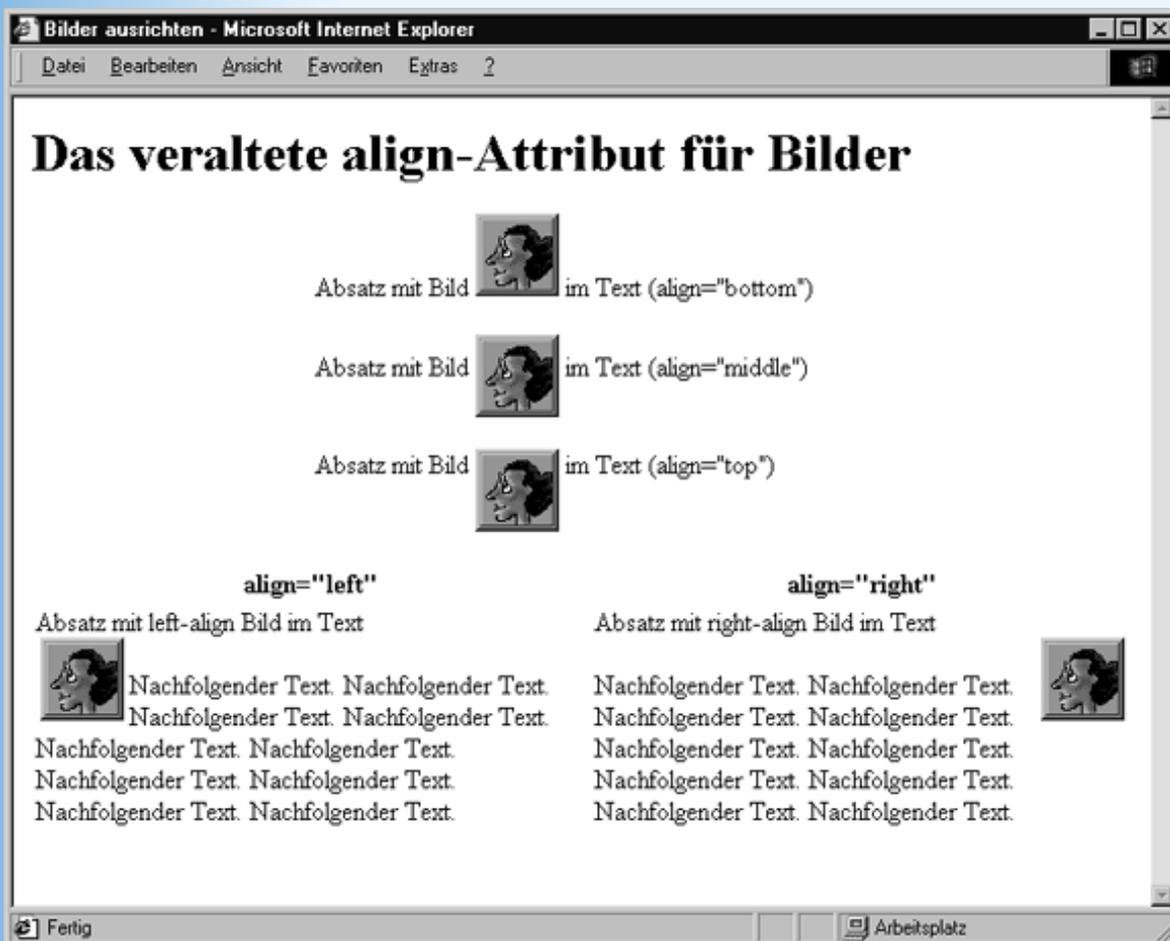


Abbildung 5.8: Traditionelle Ausrichtung von Bildern mit align

CSS sieht für die gleichen Aufgaben die Eigenschaften float und vertical-align vor (die im Übrigen auf beliebige Elemente, vertical-align allerdings nur auf Inline-Elemente, angewendet werden können).

Eigenschaft	Bedeutung
vertical-align	Ausrichtung von Inline-Elementen an der Grundlinie (baseline) der umgebenden Textzeile. Werte: baseline sub super top text-top middle bottom text-bottom <Prozentangabe>
float	Löst ein Element aus dem normalen Fluss der HTML-Elemente und positioniert es an den linken oder rechten Rand. Werte: left right none

Tabelle 5.6: Die Stileigenschaften vertical-align und float

5.7 Zusammenfassung

Heute haben Sie gelernt, wie der Browser die HTML-Elemente einer Webseite zueinander ausrichtet und wie Sie in die Positionierung eingreifen können.

Am interessantesten ist sicherlich die absolute Positionierung (position: absolute; left: 10px; top: 20px), bei der Sie explizit angeben können, an welcher Stelle ein gegebenes HTML-Element angezeigt werden soll. Weniger interessant, aber hin und wieder auch ganz nützlich, sind die relative Positionierung (position: absolute; left: 10px; top: 20px), bei der ein Element nachträglich verschoben wird, und die Positionierung mit den Stileigenschaften vertical-align und float.

Durch absolute (oder relative) Positionierung kann man HTML-Elemente überlagern. Dies ist eine ganz neue Errungenschaft, die mit HTML alleine nicht möglich war. Bei der Überlagerung von Elementen spielt die z-Reihenfolge eine große Rolle. Sie legt fest, in welcher Reihenfolge die Elemente übereinander gestapelt werden. Um selbst den z-Wert eines Elements zu definieren, verwendet man die Stileigenschaft `z-index`.

Schließlich haben wir uns noch das Clipping von HTML-Elementen angeschaut.

5.8 Fragen und Antworten

Frage:

Wenn `position: static` die Standardeinstellung ist, wozu sollte ich dann je selbst einem HTML-Element die Stileigenschaft `position: static` zuweisen?

Antwort:

In den gleichen Fällen, in denen Sie auch andere Voreinstellungen für Stileigenschaften wiederherstellen. Stellen Sie sich vor, Sie wollen abgesehen von einer Ausnahme alle Bilder (``-Elemente) einer Webseite absolut positionieren. Statt einem Dutzend oder mehr Bildern einzeln die Stileigenschaft `position: absolute` zuzuweisen, ist es sinnvoller die Stileigenschaft `position: absolute` per Stylesheet-Definition (ausnahmslos) allen Bildern zuzuweisen und dann die Positionierung für das eine auszunehmende Bild mit Hilfe einer class- oder ID-Stildefinition auf `position: static` zurückzusetzen.

Frage:

Ich möchte nicht, dass der Absatz unter einem an den linken Rand verschobenen Bild (`float: left`) rechts von dem Bild beginnt. Kann ich das verhindern?

Antwort:

Ja, indem Sie dem nachfolgenden Absatz die Stileigenschaft `clear: left` zuweisen (alternativ `clear: right`, wenn rechts von dem Absatz keine `float`-Elemente erscheinen sollen). Der Absatz beginnt dann unter dem Bild.

Frage:

Kann man die Möglichkeiten der CSS-Positionierung nicht auch dazu verwenden, Bilder zu animieren, beispielsweise mit Hilfe eines JavaScripts von links nach rechts wandern zu lassen?

Antwort:

Ja, tatsächlich gibt es viele solcher Animationen. In Kapitel 16 werden wir zeigen, wie das geht.

Frage:

Kann man zum Clipping mit Hilfe der Stileigenschaft `clip` auch noch andere Clip-Formen als `rect` verwenden?

Antwort:

Nein, derzeit gibt es nur rechteckige Clipping-Bereiche.

Frage:

Beschreibt die CSS2-Spezifikation noch andere Techniken aus CSS Positioning und Clipping?

Antwort:

Die CSS2-Spezifikation birgt noch viele interessante Möglichkeiten, wie zum Beispiel die Unterstützung verschiedener Medien, alternative Stylesheets für Anzeige und Druckausgabe, Erzeugung hierarchisch nummerierter Listen, fortgeschrittene Tabellenformatierung. Leider werden diese Möglichkeiten aber noch kaum von den Browsern unterstützt werden. Wenn Sie sich für diese Techniken interessieren, laden Sie sich die Spezifikation von der W3C-Website (<http://www.w3.org/>) herunter.

5.9 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die

Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

Quiz

1. Mit welchen Stileigenschaften kann man ein HTML-Element absolut positionieren?
2. Was ist das Besondere an der relativen Positionierung?
3. Ist die absolute Positionierung der Positionierung mit Tabellen oder Rändern (margin) grundsätzlich vorzuziehen?
4. Welche beiden Stileigenschaften dienen als Ersatz für das deprecated align-Attribute für -Bilder?

Übungen

1. Auf der Buch-CD finden Sie im Unterverzeichnis zu diesem Kapitel zwei GIF-Bilder *sonne.gif* und *erde.gif*. Erstellen Sie mit Hilfe dieser beiden Bilder eine pechschwarze Webseite, in der man die Erde sieht, wie sie die Sonne umkreist (d.h. die Erde soll halb vor der Sonne stehen).
2. Stellen Sie Abbildung 5.8 mit Hilfe der Stileigenschaften float und vertical-align nach. (Hinweis: der Seitenaufbau ist dabei nicht so wichtig, auch nicht welches Bild Sie verwenden. Wichtig ist die Demonstration der verschiedenen Positionierungsbefehle.) Freunde des Navigators 4 sind von dieser Aufgabe befreit.

❖ Kapitel Inhalt Index **SAMS** ❖ Top Kapitel ❖

1

und dienen immer noch, denn obwohl deprecated wird das Attribut von den meisten Browsern unterstützt.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

Tag 6

Fortgeschrittenes und dynamisches HTML

Heute werden wir uns einige fortgeschrittene HTML-Techniken anschauen und dabei auch schon die ersten Gehversuche in Richtung dynamisches HTML unternehmen. (Aber bitte, nicht zuviel Dynamik erwarten. Schließlich fehlen uns für echtes dynamisches HTML ja noch die Kenntnisse in der clientseitigen Skriptprogrammierung.)

Die Themen heute:

- Das <object>-Tag und seine Besonderheiten
- Einbindung von Soundeffekten
- Aufspringende Quickinfos
- Angabe einer Basisadresse für relative Hyperlinks
- Verwendung der <link>- und <meta>-Tags
- Angabe von Sprachinformationen
- Seitenübergänge im Internet Explorer
- Wie macht man Webseiten bekannt?

6.1 Multimediale Objekte einfügen

Traditionell war HTML rein textorientiert - mit der einzigen Ausnahme, dass mit Hilfe des -Tags Grafiken und Bilder eingefügt werden konnten. 1993 entwickelte die Firma Sun die Internet-Programmiersprache Java, mit der man auch Programme zur clientseitigen Unterstützung von Webseiten schreiben konnte. Also wurde der HTML-Standard um ein neues Tag, <applet> zur Einbindung von Java-Applets erweitert. Damit war aber noch lange nicht Schluss: Microsoft bestand auf Unterstützung für seine ActiveX-Elemente (Microsofts plattformabhängige Java-Applet-Alternative) und Webdesigner weltweit forderten Unterstützung für mehr und mehr multimediale Inhalte.

Die Folge war, dass Netscape das <embed>-Tag einführte, mit dem Webautoren beliebige multimediale Inhalte (Sound, Videos, etc.) in eine Webseite einfügen können, die auf der Clientseite mit Hilfe von Plug-Ins oder speziellen auf den Rechnern der Anwender installierten Programmen abgespielt werden. Microsoft übernahm das <embed>-Tag und führte zusätzlich ein <object>-Tag ein, mit dem MS Windows-verliebte Webautoren ActiveX-Steuerelemente in ihre Seiten einbauen konnten.

Um das Chaos zu komplettieren, entschloss sich das W3-Konsortium, seinen HTML 4-Standard um ein neues, generisches Multimedia-Tag zu bereichern, mit dem man beliebige Objekte in Webseiten aufnehmen kann, und nannte dieses Tag <object>-Tag.

Das <object>-Tag - es könnte alles so schön sein

Um es vorwegzunehmen: Die Unterstützung des <object>-Tags durch die großen Browser ist bestenfalls als mäßig einzustufen. Da aber zu hoffen bleibt, dass das <object>-Tag in Zukunft mehr Unterstützung erfahren und an Bedeutung gewinnen wird, werde ich Ihnen jetzt etwas darüber erzählen, wie das W3-Konsortium seinen Einsatz sieht.

Zur Einstimmung greifen wir auf Bekanntes zurück und schauen uns an, wie man mit Hilfe des <object>-Tags

Bilddateien einbinden kann. Zur Erinnerung: Mit Hilfe des - Tag, das wir nach wie vor verwenden dürfen, werden Bilddaten wie folgt eingebunden:

```

```

Mit dem <object>-Tag würde dies wie folgt aussehen:

```
<object data="kiefer.gif" type="image/gif"
        width="200" height="300">
    Kiefer, chinesische Tuschezeichnung
</object>
```

Der URL zur Bilddatei wird über das Attribut data angegeben. Da das <object>-Tag zur Einbindung der verschiedensten Multimedia-Dateien genutzt werden kann, müssen wir den Medientyp explizit angeben: type="image/gif" (siehe <ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types> für eine Liste der erlaubten Medientypen). Schließlich wollen wir noch den Text angeben, der angezeigt werden soll, wenn die Bilddatei nicht ausgewertet und dargestellt werden kann. Hier gibt es eine besonders erfreuliche Neuerung: Der alternative Text wird nicht mehr als Wert des alt-Attributs angegeben, sondern steht als HTML-Code zwischen den <object>-Tags. Das heißt, wir können nun auch komplexere und formatierte HTML-Inhalte als Alternative angeben, ja man kann sogar eine Art Prioritätenliste erzeugen:

```
<object data="kiefer.gif" type="image/gif" width="200" height="300">
    <object data="kiefer.png" type="image/png" width="200" height="300">
        Kiefer, chinesische Tuschezeichnung
    </object>
</object>
```

Hier wird zuerst versucht, ein GIF-Bild anzuzeigen. Nur wenn dies fehlschlägt, soll der Browser den Inhalt zwischen den <object>-Tags verarbeiten - in diesem Fall also das PNG-Bild laden. Misslingt auch dies, wird ein alternativer Text ausgegeben.

Bevor wir sich jedoch allzu sehr in diese Syntax verlieben, sollten Sie sich einmal anschauen, wie Ihr(e) Browser diesen Code umgeht, denn zwischen den Vorstellungen des HTML 4-Standards und der Umsetzung durch die Browser klafft hier immer noch ein ziemlicher Abgrund.

<embed> - Die Realität

Laut Standard darf und soll das -Tag zur Einbindung von Grafiken weiter neben dem <object>-Tag verwendet, zur Einbindung von Java-Applets soll das <object>-Tag verwendet werden (das <applet>-Tag wurde als »deprecated« eingestuft) und zur Einbindung von Multimedia-Inhalten muss das <object>-Tags verwendet werden, weil der HTML-Standard das <embed>-Tag der großen Browser offiziell nie anerkannt hat.

Die Realität sieht derzeit aber noch so aus, dass wir

- zur Einbindung von Grafiken und Bildern immer verwenden
- zur Einbindung von Applets augenblicklich noch <applet> vorziehen sollten, da derzeit nur der Netscape Navigator die Einbindung mit dem <object>-Tag richtig unterstützt (siehe Kapitel 15 zur Einbindung von Applets).
- zur Einbindung von multimedialen Inhalten weiterhin auf das <embed>-Tag zurückgreifen, weil die Unterstützung des <object>-Tags seitens der Browser zu dürftig und zu unterschiedlich ist.

Das <embed>-Tag

Um eine Mediendatei mit Hilfe des <embed>-Tags in eine Webseite einzubinden, brauchen Sie nur den URL

der Datei im Attribut src anzugeben:

```
<embed src="hintergrundsound.wav" />
```

Zum Abspielen der Mediendatei greift der Browser auf ein passendes Programm oder Plug-In zu. Welches Programm das richtige ist, erkennt der Browser an der Dateierweiterung.

Wenn Sie möchten, dass die Bedienfläche des Abspielprogramms angezeigt werden soll, setzen Sie das Attribut hidden auf false.

Wenn die Bedienoberfläche nicht gewünscht wird, beispielsweise beim Abspielen eines Hintergrundsounds, setzen Sie hidden auf true. Wenn es aber keine Bedienoberfläche gibt, wie kann dann das Abspielen der Mediendatei gestartet werden? Nun, zum einen mit Hilfe eines JavaScripts, zum anderen indem man das autostart-Attribut auf true setzt:

```
<embed src="hintergrundsound.wav" hidden="true" autostart="true" />
```

Soll die Datei mehrfach hintereinander abgespielt werden, weist man die Anzahl der gewünschten Wiederholungen dem Attribut loop zu (loop="true" führt zur endlosen Wiederholung).

```
<embed src="hintergrundsound.wav" hidden="true" autostart="true"
      loop="true" />
```

Schließlich kann man noch über width und height die Größe der Anzeigefläche im Browser angeben, was für Videos oder bei eingeblendetem Bedienfeld interessant ist.

Sounddateien einbinden

Als Beispiel und Testseite zur Verwendung von <object> und <embed> habe ich Ihnen folgende HTML-Seite zusammengestellt:

Listing 6.1: sound.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Songs</title>
</head>
<body>
<embed src="hintergrundsound.wav" autostart="true" hidden="true"
      loop="true" />

<h1>Play it again</h1>
<p><em>Diese Seite verfügt über einen Hintergrundsound, der endlos abgespielt
wird</em></p>
<hr />
<h2>Sounddateien über Hyperlinks aufrufen</h2>
<a href="bing.wav">Klick mich!</a>
<hr />
<h2>Sounddateien über object-Tag aufrufen</h2>
<object data="bing.wav" type="audio/wav" width="300" height="100">
  Objekt kann nicht ausgeführt werden!
</object>
</body>
</html>
```

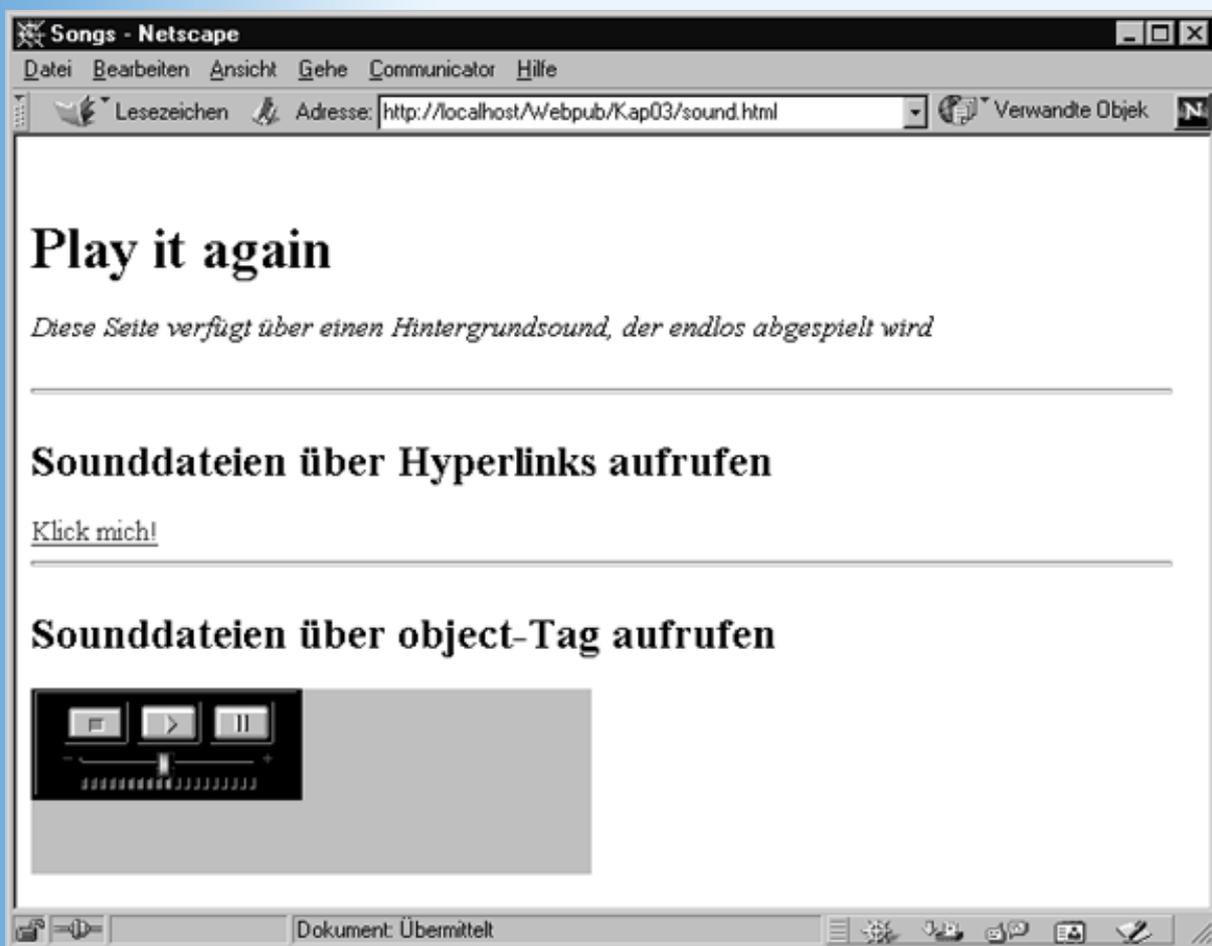


Abbildung 6.1: Sound-Testseite

Als Format für die beiden auf dieser Webseite verwendeten Sounddateien habe ich mich für WAV entschieden, da dieses Format plattform- und browserübergreifend gut unterstützt wird und Windows 9x/2000/Me standardmäßig mit einem Programm zur Bearbeitung von WAV-Dateien ausgeliefert wird (Audiorecorder).

Extension	Sound-Format
wav	WAV-Dateien. Zum Abspielen von WAV-Dateien steht für Windows/Internet Explorer der Windows Media Player und für Netscape das LiveAudio-Plug-In zur Verfügung. Unter Windows kann man WAV-Dateien mit dem Audiorecorder bearbeiten.
rm, ram	RealAudio-Dateien. Zum Abspielen von RealAudio-Dateien muss auf Seiten des Clients ein passendes Plug-In oder Abspielprogramm installiert sein. Den RealPlayer zum Abspielen von RealAudio-Dateien gibt es für alle gängigen Plattformen. Websurfer können ihn kostenlos von der Website http://www.realnetworks.com herunterladen. Webautoren können sich von der gleichen Website Aufnahmegeräte zur Erstellung von RealAudio-Dateien herunterladen (eine Basisversion gibt es kostenfrei).
mp3	MPEG-Audiodateien. Auch zum Abspielen von MPEG2-Audiodateien gibt es praktisch für jede wichtige Plattform kostenfreie Abspielprogramme (siehe beispielsweise unter www.winamp.com). Als kostengünstiges Aufnahmeprogramm kann man sich das Shareware-Tool <i>Mp3Wizard</i> herunterladen (beispielsweise von www.c3sys.demon.co.uk).

Tabelle 6.1: Gängige Sound-Formate

Der Hintergrundsound wurde mit Hilfe des <embed>-Tags eingefügt:

```
<embed src="hintergrundsound.wav" autostart="true" hidden="true"
```

```
loop="true">
```

Der zweite Sound, ein einfacher »Bing«-Ton wird mit dem <object>-Tag eingefügt. Der Internet Explorer spielt die Datei nach dem Laden sofort ab. Der Netscape Navigator wartet bis der Besucher der Website den Play-Schalter der Konsole drückt.

```
<object data="bing.wav" type="audio/wav" width="300" height="100">  
  Objekt kann nicht ausgeführt werden!  
</object>
```

Als letzte (und meist beste) Option enthält die Webseite einen Link zur *bing.wav*-Datei. Wenn der Leser dem Link nachfolgt, wird das Abspielprogramm zu der Mediendatei in einem eigenen Fenster aufgerufen.

```
<a href="bing.wav">Klick mich!</a>
```

Wenn Sie die Webseite auf Ihrem System ausführen und ein Abspielprogramm für WAV- Dateien installiert ist, sollten Sie zumindest den Hintergrundsound hören und dem Hyperlink folgen können.



Lauschen Sie ein wenig dem Hintergrundsound dieser Webseite. Gefällt er Ihnen? Lauschen Sie noch ein wenig. Spätestens nach einer halben Stunde dürften Sie überzeugt sein, dass das Einbetten von Hintergrundklängen in Webseiten (selbst wenn sie nicht endlos wiederholt werden, sondern nur bei jedem neuen Aufrufen der Seite gestartet werden) meist keine gute Idee ist. Wenn Sie den Besuchern Ihrer Webseite Soundproben anbieten wollen, tun Sie dies über Hyperlinks.

Videos

Für Videos (AVI, MPEG, Flash) gilt grundsätzlich das Gleiche wie für Sounddateien (siehe auch Kapitel 16 zur Einbindung von Flash-Animationen).

6.2 Quickinfos

Quickinfos sind kurze kontextsensitive Hilfetexte, die man bislang eher von Programmoberflächen als von Webseiten her kennt. Dort werden sie zum Beispiel als Hilfetexte zu den Schaltflächen aus den Symbolleisten eingesetzt. Verweilt der Anwender einen Moment mit der Maus über einer Schaltfläche, wird der Hilfetext zu der Schaltfläche neben der Maus eingeblendet.

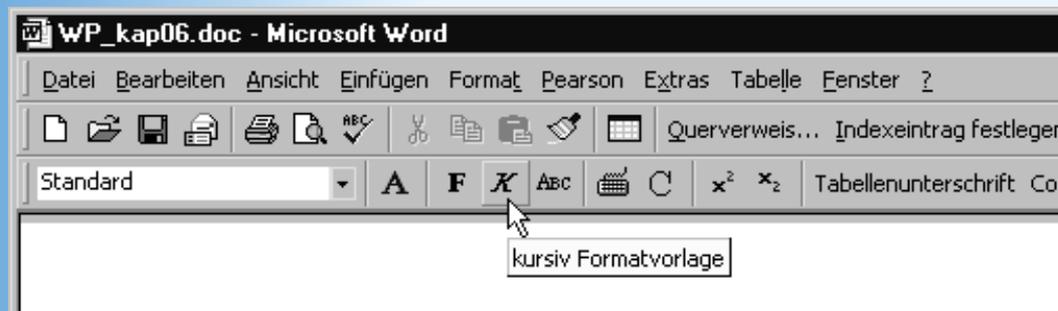


Abbildung 6.2: Quickinfos in Word

Die Möglichkeit zum Einblenden von Quickinfos gibt es mittlerweile auch in HTML. Sie brauchen lediglich das HTML-Element, für das Sie eine Quickinfo-Beschreibung anbieten wollen, um das title-Attribut zu erweitern und diesem als Wert den anzuzeigenden Text zuzuweisen:

```

```



Rücken Sie den Text für die Kurzbeschreibung nicht im HTML-Code ein (also nicht so machen, wie Sie es in obigem Quellcode-Fragment sehen). Da es sich um anzuzeigenden Text handelt, bleiben die aufeinanderfolgenden Leerzeichen erhalten. Geben Sie den Text am besten in einem Fluss ohne manuellen Zeilenumbruch ein.

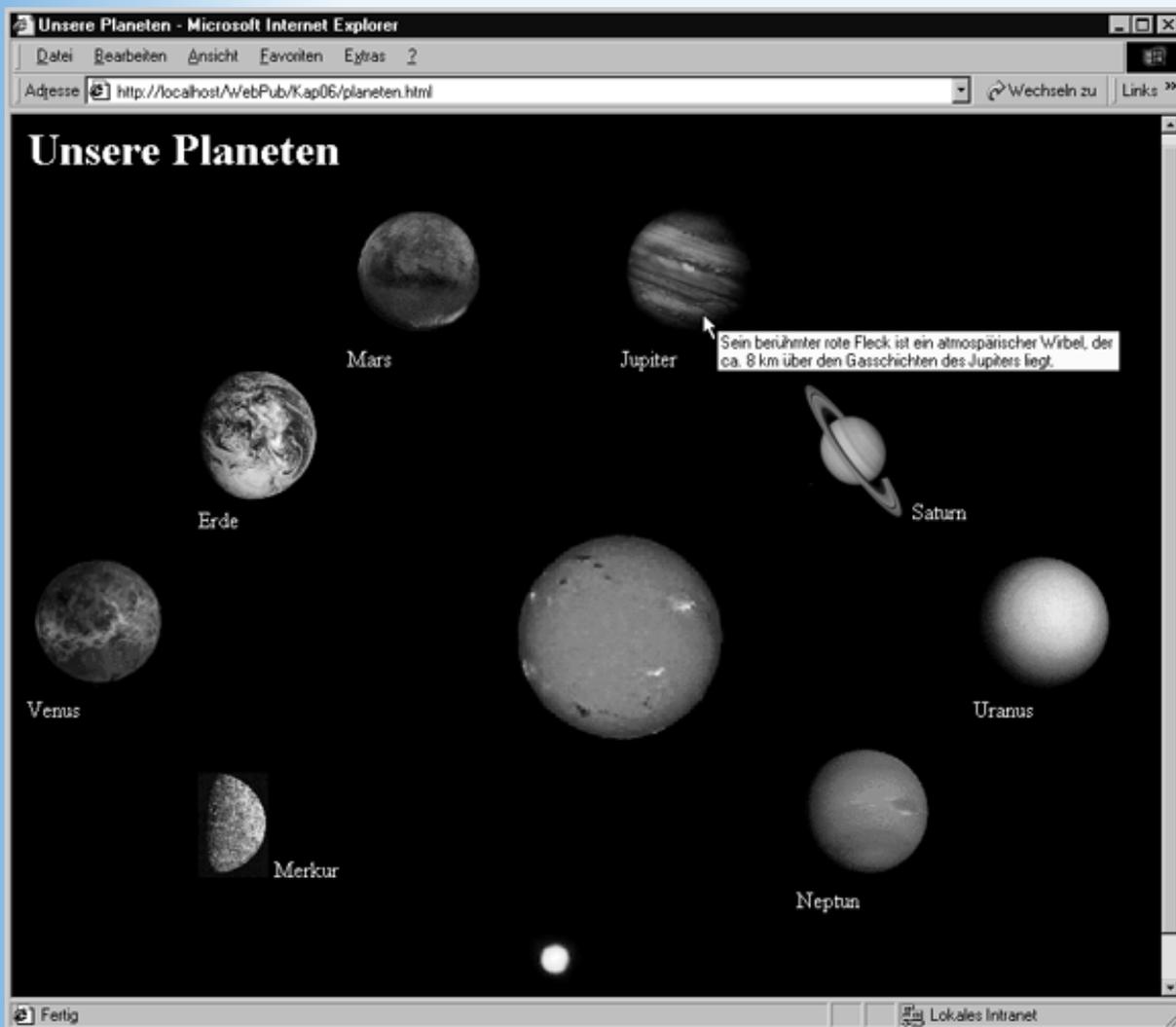


Abbildung 6.3: Kurzinformationen im Browserfenster

Quickinfos sollten aus zwei Gründen keine wichtigen Informationen enthalten:

- Nicht alle Browser unterstützen Quickinfos.
- Selbst wenn der Browser Quickinfos unterstützt, kann es sein, dass der Leser der Webseite die Quickinfos nicht abrufen kann.

Nehmen wir zum Beispiel die Webseite aus Abbildung 6.3. Die Planetensymbole sollen als grafische Hyperlinks zu den einzelnen Planeten-Webseiten führen, wo der Besucher der Website detaillierte Informationen über die Planeten findet. Man könnte nun auf die Idee kommen, sich die »störenden« Untertitel zu den Planeten zu

sparen und die Planetennamen stattdessen als Quickinfos vorsehen. Leser, die sich nicht sicher sind, welches Planetensymbol für welchen Planeten steht, brauchen dann nur die Quickinfos abzurufen. Dieser Ansatz hat aber seine Haken und Ösen. Zum einen gibt es derzeit viel zu viele Websurfer, die diese für die Navigation wichtige Information mangels Browserunterstützung überhaupt nicht abrufen können, zum anderen dauert das Abrufen der Quickinfos viel zu lange - insbesondere dann, wenn man den gewünschten Planeten erst beim n-ten Male findet. Besser ist es meist, wie im Beispiel demonstriert, die für die Navigation relevanten Informationen gut sichtbar anzuzeigen und als Quickinfo weniger wichtige Zusatzinformationen zu liefern - etwa wie im vorliegenden Beispiel kleine Informations-Appetithappen, die den Websurfer animieren, sich die betreffende Seite doch einmal anzusehen.

6.3 Metainformationen und Header-Tags

Erinnern Sie sich noch an den Anfang des zweiten Tages, als wir Sie mit den Formalitäten des HTML-Grundgerüsts und an den Browser gerichteten Informationen wie `<!DOCTYPE>` oder `<title>` geplagt haben, während Sie ungeduldig darauf warteten, dass wir unser HTML-Dokument mit sichtbarem Inhalt füllen? Seitdem haben wir uns fast nur noch mit dem `<body>`-Bereich und der Darstellung der anzuzeigenden HTML-Inhalte beschäftigt. Jetzt, am Ende unseres Streifzugs durch die HTML-Elemente, wenden wir uns noch einmal dem Header-Abschnitt zu. Und mit der Abgeklärtheit und Gelassenheit des (mittlerweile) erfahrenen Webdesigners, der weiß, wie er seine gestalterischen Mittel nutzen und einsetzen kann, wollen wir uns anschauen, was uns der Header-Abschnitt zu bieten hat.

Um es gleich vorweg klarzustellen: Was im Header-Bereich des HTML-Dokuments steht, gehört nicht zum eigentlichen Inhalt des HTML-Dokuments und wird vom Browser folglich nicht im Anzeigebereich des Browser-Fensters dargestellt! Das heißt nun nicht, dass die im Header-Bereich stehenden Informationen gar nichts mit der Darstellung im Browser-Fenster zu tun haben müssen, denn auch wenn sie selbst nicht als Teil der Webseite angezeigt werden, können sie doch auf die Darstellung im Browser rückwirken oder in anderer Form angezeigt werden (man denke nur an die Einbindung von Stylesheets mit `<style>` oder an das `<title>`-Tag).

Aber auch die Header-Informationen, die nichts mit der Darstellung der Webseite im Browser zu tun haben, können für den Webdesignern interessant sein. Ihnen wird in diesem Abschnitt sogar unser Hauptaugenmerk gelten.

Die HTML-Tags für den Header-Bereich

Tabelle 6.2 bietet einen Überblick über die HTML-Tags des Header-Bereichs, von denen Sie einige bereits kennen gelernt haben.



Mit Ausnahme von `<title>` sind alle HTML-Tags des Header-Bereichs optional.

Tag	Beschreibung
<code><title></code>	Titel des HTML-Dokuments, der üblicherweise in der Titelleiste der Browser angezeigt wird (siehe Kapitel 2.2.2).
<code><base></code>	Basisadresse des HTML-Dokuments, das zur Vervollständigung relativer URLs verwendet wird (siehe unten).
<code><meta></code>	Tag zur Aufnahme optionaler Metainformationen. Dieses Tag ist sehr vielseitig einsetzbar, siehe Abschnitt 6.3.4.

<code><script></code>	Tag zur Definition von Skripten (siehe Kapitel 8).
<code><style></code>	Tag zur Definition von Stylesheets (siehe Kapitel 4).
<code><link></code>	Tag zur Einbindung externen Informationen (siehe unten).
<code><object></code>	Tag zur Einbindung globaler Objekte in Frameseiten.

Tabelle 6.2: HTML-Tags des Header-Bereichs

<base> - Die Basisadresse

Wenn Sie auf einer Webseite einen URL als Link zu einer Webressource angeben, haben Sie die Wahl zwischen

- einem absoluten URL oder
- einem relativen URL.

Ein absoluter URL ist eine vollständige Webadresse, wie zum Beispiel:

```
http://www.meinedomaene.de/unterverzeichnis/webseite.html
```

Er besteht aus

- einer Protokollangabe (für Webseiten `http://`)
- der Hostangabe, die üblicherweise aus drei Teilen besteht: dem Präfix `www` (das allgemein zur Kennzeichnung von Webserver verwendet wird), dem sekundären Domännennamen (hier `meinedomaene`) und dem primären Domännennamen (hier `de`).
- einer Pfadangabe (hier `/unterverzeichnis/`)
- dem Namen der gesuchten Ressource (hier `webseite.html`)

Man kann URLs auf Webseiten aber auch als relative URLs angeben. Statt der vollständigen Webadresse gibt man dabei nur den Pfad an, der vom Verzeichnis des aktuellen Webdokuments zu der gewünschten Ressource führt. Oder man gibt im Header- Bereich mit Hilfe des `<base>`-Tags eine Basisadresse an, auf die sich alle relativen URLs der aktuellen Webseite beziehen sollen (siehe auch Kapitel 2.6).

```
<head>
  <title>Hallo</title>
  <base href="www.einServer.de/verz" />
</head>
```

Relative versus absolute URLs

Grundsätzlich sollte man alle URLs, die auf Webseiten und Ressourcen verweisen, die im Verzeichnissystem des eigenen Webs stehen, als relative URLs angeben und absolute URLs nur für solche URLs verwenden, die auf Webseiten oder Ressourcen außerhalb des eigenen Webs verweisen.

Warum? Weil Webs mit relativen URLs bequem zu verschieben sind! Schauen wir uns einmal folgende Webseite an:

Listing 6.2: kiefer.html - Webseite mit relativen und absoluten URLs

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
```

```

<title>Die Kiefer</title>
</head>
<body style="background-image: url('../images/kiefer.gif');
        -background-repeat: no-repeat;
        -margin-left: 200; margin-top: 210">
<h1>Die Kiefer in der chinesischen Malerei</h1>
<p>In der traditionellen chinesischen Malerei gehört die Kiefer zu den am
häufigsten dargestellten Bäumen überhaupt. Diese Vorrangstellung hat sie ihrer
Genügsamkeit und ihren immergrünen Nadeln zu verdanken, die sie in der
chinesischen Vorstellung zu einem Symbol für langes Leben und Beständigkeit
werden ließen. So gehört die Kiefer auch zu den <em>drei Freunden im
Winter</em> &#150; den langes Leben symbolisierenden Pflanzen Kiefer,
<a href="http://localhost/demoweb/bambus.html">Bambus</a> und
<a href="pflaume.html">Pflaume</a>. ...</p>
<p style="text-align:right">Besuchen Sie auch unseren
<a href="www.chinsponsor.com">chinesischen Sponsor.</a></p>
</body>
</html>

```

Diese Webseite, die Sie im Übrigen als Teil eines kleinen Testwebs, namens *DemoWeb*, auf der Buch-CD finden, enthält zwei relative und zwei absolute URLs (sicherlich haben Sie keine Mühe, diese vier URLs auszumachen und zuzuordnen). Zu dieser Webseite gehört die Webstruktur aus Abbildung 6.4.

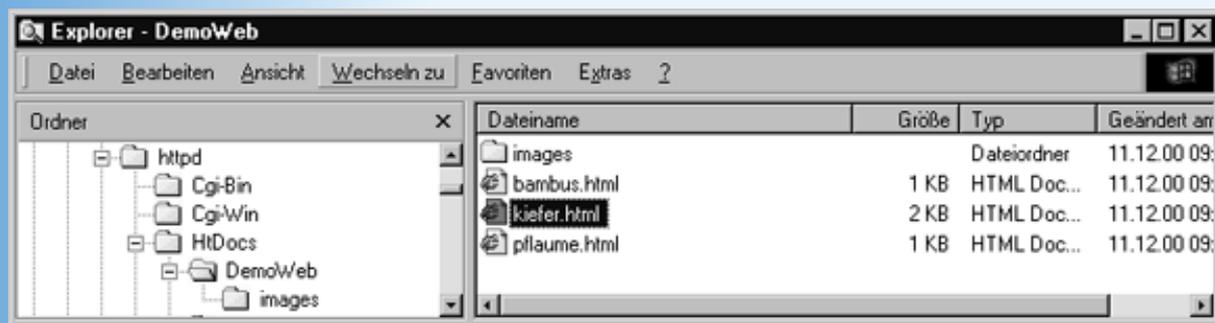


Abbildung 6.4: Webverzeichnisystem zu DemoWeb

Unter der Voraussetzung, dass der Webservername localhost mit dem Verzeichnis *c:\httpd\htdocs* verbunden ist, kann man die Webseite *kiefer.html* über *http://localhost/demoweb/kiefer.html* aufrufen.

Gehen wir nun den einzelnen URLs der Webseite *kiefer.html* nach.

```

<body style="background-image: url('../images/kiefer.gif');

```

Dieser relative URL lädt das Hintergrundbild *kiefer.gif* der Webseite. Der URL gibt an, dass die Bilddatei *kiefer.gif* im Verzeichnis *images* zu suchen ist, das ein direktes Unterverzeichnis des Verzeichnisses ist, in dem die aktuelle Webseite abgespeichert ist.



Die Verzeichnisangabe ./ steht immer für das aktuelle Verzeichnis. Die Angabe ../ bezieht sich auf das übergeordnete Verzeichnis und kann aneinandergereiht werden, um im Verzeichnispfad beliebig hoch zu wandern (../..../gibt_es_dieses_verzeichnis_noch.html).

```

<a href="http://localhost/demoweb/bambus.html">Bambus</a>

```

Dies ist ein absoluter URL (statt der Domänenangabe steht hier der Name des lokalen Webservers), der auf eine Webseite namens *bambus.html* verweist, die im gleichen Verzeichnis wie die aktuelle Webseite *kiefer.html* steht. Dies ist statthaft, hat allerdings auch Nachteile, wie wir gleich sehen werden.

```
<a href="pflaume.html">Pflaume</a>
```

Dies ist ein relativer URL, der auf eine Webseite namens *pflaume.html* verweist, die im gleichen Verzeichnis wie die aktuelle Webseite *kiefer.html* steht.

```
<p>Besuchen Sie auch unseren <a href="www.chinsponsor.com">chinesischen  
Sponsor.</a></p>
```

Dies ist ein absoluter URL, der auf eine Webseite außerhalb unseres eigenen Webs verweist.

So weit, so gut. Wenn Sie die Webseite laden, wird das Hintergrundbild wie gewünscht aus dem Unterverzeichnis *images* geladen und per Klick kann man den Links zu den anderen Webseiten nachgehen.

Problematisch wird es, wenn das Webverzeichnis verschoben oder umbenannt wird - also wenn Sie das Verzeichnis *C:\httpd\htdocs\demoweb* in *C:\httpd\htdocs\chinmalerei* umbenennen oder von Ihrem lokalen Webserver (*localhost*) auf einen Internetrechner hochladen. Während der absolute URL auf die externe Website und die relativen URLs auch nach einer solchen Änderung weiterhin korrekt sind (sofern Sie nicht auch noch Namen und Lage der Unterverzeichnisse ändern), stimmt der absolute URL auf die webinterne Webseite *bambus.html* nicht mehr. Daher die Faustregel, auf Ressourcen, die in der Verzeichnisstruktur des eigenen Webs liegen, grundsätzlich über relative URLs zu verweisen.



*Wenn Sie zur Probe das Verzeichnis *demoweb* in *demoweb2* umbenennen und beim Laden in den Webbrowser feststellen sollten, dass Sie die Datei *bambus.html* nach wie vor über den Link `Bambus` ansteuern können, liegt dies daran, dass Sie dem Link schon einmal gefolgt sind und Ihr Browser die Datei noch im Cache hat. Lassen Sie die Anzeige dann aktualisieren und drücken Sie gleichzeitig je nach Browser die Taste (Strg) oder (Shift).*

Relative Links sind auch wichtig, wenn Sie die Dateien eines Webs als ZIP-Datei zum Herunterladen anbieten. Wenn Sie absolute URLs zum Verbinden der Webseiten verwenden, hat der Leser Ihrer Webseiten das Problem, dass er zwar alle Webseiten auf seiner Festplatte vorliegen hat (und die Dateien bei richtiger Komprimierung und Entpackung auch in den korrekten Unterverzeichnissen liegen), dass er aber die Dateien nur über den **Datei/Öffnen**-Befehl seines Browsers laden kann, weil die Links zwischen den Dateien auf die Dateien im Internet verweisen.

Wann gibt man eine Basisadresse vor?

Die Vorteile der relativen Links gehen verloren, wenn man eine Basisadresse vorgibt, weil sich die relativen Links danach nicht mehr auf das Verzeichnis des aktuellen Dokuments, sondern auf die Basisadresse beziehen. Um bei unserem Beispiel von oben zu bleiben:

Wenn Sie Listing 6.2 um die Angabe einer Basisadresse erweitern, die auf das Verzeichnis des Beispielwebs verweist:

```
<head>
```

```
<title>Die Kiefer</title>
<base href="http://localhost/demoweb/" />
</head>
```

bleibt zunächst alles beim alten (da die Basisadresse dem Verzeichnis der Datei *kiefer.html* entspricht). Wenn Sie das Webverzeichnis aber umbenennen oder auf einen anderen Webserver portieren, stimmt kein relativer Pfad mehr.

Wozu also überhaupt eine Basisadresse vorgeben?

Die Angabe einer Basisadresse lohnt sich immer dann, wenn alle (oder zumindest ein großer Teil) der Links einer Webseite auf Ressourcen außerhalb der aktuellen Website liegen.

Nehmen wir an, Sie haben eine private Website auf dem Webserver eines Freundes veröffentlicht. Nun bietet sich Ihnen die Möglichkeit, Ihre Website auch auf dem Webserver eines anderen Internet-Providers zu veröffentlichen. Aber vielleicht können Sie auf dessen Rechner keine Unterverzeichnisse anlegen oder keine CGI-Programme installieren oder es steht Ihnen einfach nicht genügend Festplattenspeicher zur Verfügung. In so einem Fall könnten Sie es so machen, dass Sie lediglich die zentrale Homepage Ihres Webs auf den anderen Webserver kopieren und in der kopierten Homepage als Basisadresse den URL Ihrer Website auf dem Webserver Ihres Freundes angeben. Webbesucher finden Ihre Website dann auch auf dem Webserver des anderen Internet- Providers, und Zugriffe von der Homepage auf die anderen Seiten des Webs werden automatisch zu Ihrem Hauptwebserver umgeleitet.¹

Eine andere Möglichkeit wäre, dass Sie von einer Webseite verstärkt auf Ressourcen (beispielsweise Bilder oder Sounddateien) zugreifen, die von einem anderen Webserver stammen. In so einem Fall könnte man sich das Eintippen der absoluten URLs zu den externen Ressourcen dadurch erleichtern, dass man den URL des externen Webserver als Basisadresse festlegt.

Schließlich kann man für das `<base>`-Tag wie für das Anker-Tag `<a>` ein `target`-Attribut festlegen. Auf diese Weise kann man mit einem Schlag für alle Link-Elemente einer Webseite (`<a>`, `<link>`, `<area>`, `<form>`), für die kein eigenes `target`-Attribut angegeben wird, einen Zielbereich (Target) festlegen - eine Option, die vor allem für Webseiten, die in Frames angezeigt werden, interessant ist (siehe Kapitel 3.2).



Wenn Sie mit dem `<base>`-Tag eine Basisadresse vorgeben, muss das `<base>`-Tag vor allen HTML-Elementen aufgeführt werden, die auf externe Quellen verweisen - also beispielsweise vor irgendwelchen `<link>`-Tags.

<link> - Beziehungen zwischen Dokumenten



Das `<link>`-Tag des `<head>`-Bereichs hat nichts mit dem `<a>`-Tag aus dem `<body>`-Bereich zu tun.

In Kapitel 4.4 sind wir schon einmal über das `<link>`-Tag des Header-Bereichs gestolpert. Dort ging es darum, wie man externe Stylesheets, die in eigenen Dateien stehen, mit einem Webdokument verbinden kann:

```
<link href="meinstyle.css" rel="stylesheet" type="text/css" />
```

Hier sehen wir bereits die beiden wichtigsten Attribute des <link>-Tags:

- href, das den URL der Datei enthält, zu der eine Verbindung hergestellt werden soll
- rel, das anzeigt, welche Art von Beziehung (englisch »relation«) zwischen den beiden Dokumenten besteht.
- (type, ist nicht ganz so wichtig. Es beschreibt wie üblich die Art von Inhalt, die von dem verbundenen Dokument zu erwarten ist).

Das ist in groben Zügen schon alles, worum es beim <link>-Tag geht. Man gibt den URL zu einem anderen Dokument an, beschreibt über das rel-Attribut die Art der Beziehung, die zwischen dem aktuellen HTML-Dokument und dem LINK-Dokument besteht, und hofft, dass der Browser (oder irgendein anderes Programm, das das HTML-Dokument liest) dem Link folgt und weiß, was er (es) mit dem zugeordneten Dokument anfangen soll.

Leider ist aber gerade letzteres oft genug nicht der Fall, und so beschränken wir uns darauf, kurz anzuschneiden, welche Form von Unterstützung sich der HTML-Standard für sein <link>-Tag wünschen würde. Entscheidend ist dabei stets der Wert des rel-Attributs, für das in der HTML-Spezifikation eine Reihe von möglichen Werten vordefiniert sind. Einige dieser Werte werden wir uns jetzt anschauen.

Alternative Webseiten

Mit Hilfe der rel-Beziehung »alternate« kann man auf alternative Darstellungen für die aktuelle Webseite hinweisen. So könnte eine spanische Suchmaschine darauf hinweisen, dass es die gleiche Webseite auch in Spanisch gibt (hreflang="es"), oder einen Browser für Blinde zum Link einer Webseite mit Audiounterstützung führen (media="braille").

```
<link title="El pino" type="text/html"
      rel="alternate" hreflang="es"
      href="http://webserver.com/meinweb/spanisch.html" />
```



Die wichtigsten Sprachcodes sind en (Englisch), en-US (Amerikanisch), fr (Französisch), de (Deutsch), it (Italienisch), nl (Holländisch), el (Griechisch), es (Spanisch), pt (Portugiesisch), ar (Arabisch), he (Hebräisch), ru (Russisch), zh (Chinesisch), ja (Japanisch), hi (Hindi), ur (Urdu) und sa (Sanskrit).

Die wichtigsten Medientypen sind screen (normaler Grafik-Bildschirm), tty (Terminal), tv (TV-Bildschirm), projection (Projektor), handheld (kleine Bildschirme wie bei PalmPilots), print (Drucker), braille (für Blinde), aural (mit Sprachengenerator) und all (Alle).

Navigation und Dokumentenhierarchie

Verschiedene rel-Beziehungen dienen dazu, Informationen über den hierarchischen Aufbau eines Webs zu liefern.

rel-Beziehung	Intention
start	Ist dazu gedacht, das Startdokument (Home) eines Webs anzuzeigen.

next	Verweist auf das »nächste« Dokument (beispielsweise die Webseite mit dem nächsten Kapitel, wenn das Web wie ein Buch organisiert ist).
prev (oder previous)	Verweist auf das »vorangehende« Dokument (beispielsweise die Webseite mit dem vorangehenden Kapitel, wenn das Web wie ein Buch organisiert ist).
contents (oder toc)	Verweist auf das Inhaltsverzeichnis
index	Verweist auf den Index
glossary	Verweist auf den Glossar
appendix	Verweist auf den Anhang
help	Verweist auf die Hilfe

Tabelle 6.3: rel-Beziehungen

Ursprünglich ging man wohl davon aus, dass die Browser aus diesen <link>-Tags automatisch eine Navigationsleiste aufbauen. Daraus ist leider nichts geworden. So definiert selbst die HTML-Spezifikation neben den entsprechenden <link>-Tags eine eigene Navigationsleiste:

```

...
<head>
  <title>The global structure of an HTML document</title>
  <link rel="previous" href="../types.html" />
  <link rel="next" href="dirlang.html" />
  <link rel="contents" href="../cover.html#toc" />
  ...
</head>
<body>
<div class="navbar" align="center">&nbsp;
  <a href="../types.html">previous</a> &nbsp;
  <a href="dirlang.html">next</a> &nbsp;
  <a href="../cover.html#minitoc">contents</a> &nbsp;
  <a href="../index/elements.html">elements</a> &nbsp;
  <a href="../index/attributes.html">attributes</a> &nbsp;
  <a href= "../index/list.html">index</a>
  ...

```

Eine weitere interessante Möglichkeit, die meines Wissens kaum (wenn überhaupt) Unterstützung fand, ist die Vorgabe einer bevorzugten Navigationsreihenfolge mit Hilfe der Wert start, next und prev. Ein Browser könnte die next- und prev-Links beispielsweise dazu nutzen, die betreffenden Seiten im Hintergrund zu laden und dadurch die eigentliche Navigation zu beschleunigen. Beim Drucken könnten die Seiten eines Webs beginnend mit dem start-Dokument Seite für Seite automatisch ausgegeben werden, indem der Browser den next-Links folgt und die aufgerufenen Seiten nacheinander an den Drucker schickt.

Schließlich können die Informationen aus den <link>-Tags von Suchmaschinen genutzt werden, um die Startseite eines Webs oder eine anderssprachige Version einer Webseite ausfindig zu machen.



Mit Hilfe eines »Profils« können Sie eigene Beziehungen (rel-Werte) definieren. Wer mehr darüber wissen möchte, sei auf die HTML-Spezifikation, Stichwort profile verwiesen.

Metainformationen

Mit dem interessantesten HTML-Tag für den Header-Bereich ist das <meta>-Tag. Mit seiner Hilfe kann der Autor einer Webseite die verschiedensten Daten und Informationen zu einer Webseite in der Webseite selbst ablegen. Alle Informationen werden dabei als Name/ Wert-Paare verpackt. Der »Name« gibt an, um welche Information es sich handelt und wird mit Hilfe des name-Attributs spezifiziert. Der »Wert« ist die eigentliche Information und wird über das content-Attribut angegeben.

Wenn Sie also sich selbst als Autor einer Webseite in deren Header-Bereich verewigen wollten, würden Sie schreiben:

```
<meta name="author" content="ihr name">
```

Interessant ist, dass der HTML-Standard keine Liste von verbindlichen Namen vorgibt. Es gibt allerdings eine Reihe von weit verbreiteten Namen mit mehr oder weniger fester Bedeutung.

META-Tag	Bedeutung
<meta name="author" content="..." />	Autor der Webseite
<meta name="copyrights" content="..." />	Hinweis auf Urheberrechte
<meta name="generator" content="..." />	Software, mit der die Webseite generiert wurde.
<meta name="description" content="..." />	Kurze Beschreibung der Seite und ihres Inhalts. Suchmaschinen zeigen diesen Text üblicherweise auf den generierten Trefferseiten neben oder unter dem URL der Webseite an.
<meta name="keywords" content="..." />	Stichwörter, unter denen Ihre Webseite von den diversen Suchmaschinen gefunden werden soll. Die einzelnen Stichwörter werden durch Kommata getrennt.
<meta name="rating" content="..." />	FSK ¹ für Webseiten. Als Wert für content stehen zur Auswahl: general (ohne Altersbeschränkung) safe for kids (für Kinder geeignet) 14 years (ab 14 Jahren) restricted (mit Einschränkungen) matured (nur für Erwachsene)

Tabelle 6.4: Name/Content-Paare für META-Tags Beziehungen

1 Freiwillige Selbstkontrolle

Simulierte HTTP-Header

Name/Wert-Paare können aber nicht nur mit dem name-Attribut spezifiziert werden. Alternativ kann man den Namen als Wert des http-equiv-Attributs angeben. Dies hat den besonderen Nebeneffekt, dass der Webserver, auf dem die Webseite steht, aus den Angaben im <meta>-Tag ein HTTP-Headerfeld erzeugt und diesen als Teil des HTTP- Antwortheaders an den Browser schickt.

HTTP-Antwortheader

Webdokumente werden gemäß dem HTTP-Protokoll über das Internet geschickt. Dieses Protokoll sieht vor, dass jedem verschickten Datenpaket ein Header vorangeht. Als Websurfer oder Webautor merken wir üblicherweise nur wenig von diesen HTTP-Headern (nicht zu verwechseln mit dem <head>-Abschnitt unserer Webseiten), da sie automatisch vom Webserver erzeugt und vom Webbrowser verarbeitet werden. Wenn Sie beispielsweise auf einen Hyperlink zu einer Webseite klicken, ermittelt der Webbrowser den Webserver, auf dem die Webseite abgelegt sein soll und schickt diesem eine Anfrage (mit vorangehendem HTTP-Anfrageheader). Der Webserver sucht in seiner Verzeichnisstruktur nach der Webseite. Kann er sie finden, stellt er ihr einen passenden HTTP-Antwortheader voran und sendet sie samt Header an den Browser. Ein typischer Antwortheader könnte wie folgt aussehen:

```
HTTP/1.0 200 OK
Date: Sun, 03 Dec 2000 11:34:52 GMT
Server: Apache/1.3.14
Content-type: text/html
Content-length: 1284
Last-modified: Tue, 05 Dec 2000 00:45:46 GMT
```

Wie kann man sich dieses Mechanismus zunutze machen?

Nun, zum einem können Sie echte HTTP-Headerfelder simulieren, ja sogar HTTP-Headerfelder, die vom Webserver gesendet werden, überschreiben - was allerdings bedeutet, dass man auf diese Weise auch großen Schaden anrichten kann.



Hüten Sie sich den content-type- oder content-length-Headerfelder zu überschreiben.

Caching

Viele Firmen stellen Proxy-Server zwischen ihre Rechner und das Internet, um den Zugriff auf Webseiten effizienter zu gestalten. Der Proxy fungiert dabei als Cache, in dem Webseiten zwischengespeichert werden. Fordert ein Webbrowser eine Webseite an, die sich im Cache des Proxy befindet, braucht die Anfrage nicht ins Internet weitergeleitet zu werden, sondern kann direkt vom Proxy beantwortet werden. So spart man Zeit und Netzwerkkapazitäten, nimmt aber dafür in Kauf, dass die gelieferten Webseiten unter Umständen nicht mehr aktuell sind. Das HTTP-Protokoll sieht verschiedene Wege vor, wie man sicherstellen kann, dass das Caching möglichst effizient abläuft, ohne allzu sehr auf Kosten der Aktualität der Webseiten zu gehen. In diese Mechanismen können wir eingreifen.

Eine Möglichkeit ist, dass Caching einer Webseite gänzlich zu unterbinden. Dazu setzt man das HTTP-Headerfeld Cache-control auf no-cache:

```
<meta http-equiv="Cache-control" content="no-cache" />
```



Das Headerfeld Cache-control ist erst sei HTTP 1.1 definiert. In HTTP 1.0 setzt man stattdessen

das Headerfeld `pragma` auf `no-cache`.

Eine andere Möglichkeit besteht darin, ein Ablaufdatum anzugeben.

```
<meta http-equiv="expires" content="Sat, 30 Nov 2001 10:55:21 GMT" />
```

In diesem Fall wird sichergestellt, dass der Browser sich nach Ablauf des angegebenen Datums bemüht, eine aktuelle Version des Dokuments zu bekommen.² (Was im Übrigen nicht bedeutet, dass das Dokument danach unbedingt vom Webserver geladen werden muss. Oft genügt es, wenn der Browser (oder der Proxy) beim Webserver nachfragen, ob es eine aktuellere Version des Dokuments gibt. Falls nein, braucht der Webserver statt des Dokuments nur eine Bestätigung zu schicken, dass weiterhin das gecachte Dokument verwendet werden kann.)

Will man erreichen, dass eine Webseite gecacht, aber trotzdem stets auf Aktualität geprüft wird, stattet man sie mit einem Ablaufdatum aus, das in der Vergangenheit liegt oder setzt das folgende Headerfeld:

```
<meta http-equiv="Cache-control" content="must-revalidate" />
```



Verwechseln Sie das Caching nicht mit der Liste der zuletzt angezeigten Webseiten. Wenn Sie sich in der Liste der zuletzt angezeigten Webseiten zurückbewegen, rekonstruiert der Browser die Webseiten üblicherweise so, wie Sie sie zuvor gesehen haben. Sinn dieser History-Liste ist es gemeinhin, bereits gesehene Inhalte identisch wiederherzustellen und nicht, bereits besuchte Webseiten neu abzurufen.

Client-Side Pull

Zum anderen können Sie mit Hilfe des `http-equiv`-Attributs Headerfelder an den Browser senden, die der Webserver gar nicht kennt (weil sie nicht zur HTTP-Spezifikation gehören), die von den Browsern aber wohl verstanden werden.

Der wohl meist eingesetzte und am breitesten unterstützte unter diesen Headerfeldern ist der von Netscape eingeführte CGI-Antwortheadfeld `Refresh`, mit dem man vom Browser aus (Client-Seite) Webseiten vom Server herunterziehen (Pull) kann. Nun werden Sie einwenden, dass dies nichts Besonderes, sondern genau das ist, was beim Anklicken eines Hyperlinks passiert. Das Besondere am Client-Side Pull ist, dass hierzu keine Benutzerinteraktion erforderlich ist, das heißt, der Browser fordert ganz automatisch die gewünschte Seite vom Webserver an.

Das zugehörige META-Tag gibt es in zwei Formen:

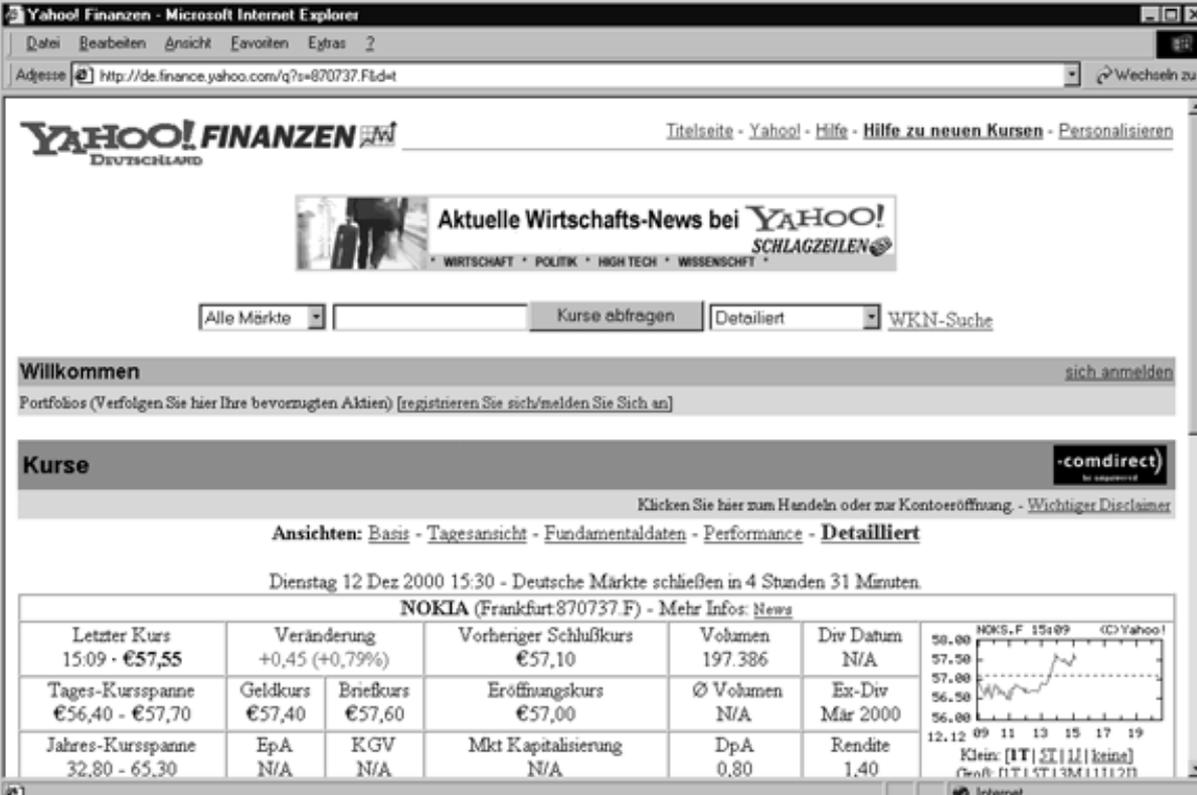
```
<meta http-equiv="refresh" content="10" />
<meta http-equiv="refresh"
      content="10; url=http://www.server.de/meinweb/start.html" />
```

Im einfachsten Fall besteht der Inhalt des `Refresh`-Headerfelds aus einem einzelnen Zahlenwert, der angibt, nach wie vielen Sekunden die aktuelle Seite (das heißt die Seite, die das META-Tag enthält) vom Webserver neu geladen werden soll.

Will man nach Ablauf der vorgegebenen Frist nicht die aktuelle, sondern irgendeine andere Webseite herunterladen, gibt man zusätzlich den URL dieser Webseite an.

Automatische Aktualisierung

Die automatische Aktualisierung wird immer dann eingesetzt, wenn eine Webseite Informationen enthält, die ständigen Veränderungen unterworfen sind. Ein typisches Beispiel sind Webseiten mit Aktienkursen.



The screenshot shows the Yahoo! Finance website for Nokia (NOKIA) on December 12, 2000. The page displays the following data:

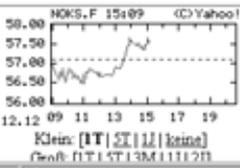
NOKIA (Frankfurt 870737 F) - Mehr Infos: News						
Letzter Kurs 15:09 - €57,55	Veränderung +0,45 (+0,79%)		Vorheriger Schlusskurs €57,10	Volumen 197.386	Div Datum N/A	
Tages-Kursspanne €56,40 - €57,70	Geldkurs €57,40	Briefkurs €57,60	Eröffnungskurs €57,00	Ø Volumen N/A	Ex-Div Mär 2000	
Jahres-Kursspanne 32,80 - 65,30	EpA N/A	KGV N/A	Mkt Kapitalisierung N/A	DpA 0,80	Rendite 1,40	

Abbildung 6.5: Yahoo-Seite mit aktuellen Börsenkursen

So werden die Yahoo-Finanzseiten zu den Börsenkursen alle 15 Minuten (900 Sekunden) aktualisiert:

```
<html>
<head>
  <title>Yahoo! Finanzen</title>
  <meta http-equiv=Refresh content=900 />
</head>
...
```



Die Yahoo-Seiten sind schon etwas älter und folglich etwas laxer im Umgang mit den neueren HTML-Empfehlungen. Stören Sie sich also nicht daran, dass keine HTML-Version deklariert ist und die Attributwerte nicht in Anführungszeichen stehen.

Die automatische Aktualisierung einer Webseite (die nicht in allzu kurzen Intervallen erfolgen sollte, sonst legt man Browser und Server lahm) macht natürlich nur dann einen Sinn, wenn auch die Inhalte der Webseite in regelmäßigen Abständen aktualisiert werden. Da man als Webadministrator seine Zeit aber nicht unbedingt damit verbringen möchte, ständig die eigenen Webseiten auf dem Server zu aktualisieren, muss man auch diesen Prozess irgendwie automatisieren. Hierfür bieten sich Server-Side Includes (siehe Kapitel 14) oder CGI-Programme (siehe Kapitel 17) an.

Automatische Weiterschaltung

Beim Surfen durchs Web stößt man immer wieder auf Seiten, die den Websurfer höflich darauf hinweisen, dass er hier falsch ist, und ihn kurzerhand zu einer anderen Website führen. Diese automatische Weiterschaltung kann ebenfalls mit dem Refresh-Headerfeld einrichten. Man muss lediglich neben der Ablauffrist noch den anzusteuernenden URL angeben:

```
<meta http-equiv="refresh"
      content="10; url=http://www.server.de/meinweb/start.html" />
```

Moderne Webs, die vor der eigentlichen Startseite (Homepage) eine Eingangsseite (Portal) stehen haben, können das Refresh-Headerfeld dazu benutzen, den Webbesucher automatisch von der Eingangsseite zur Startseite des Webs zu führen - obwohl ein gut sichtbarer Link hier meist die bessere Alternative ist.

Schließlich kann man mit Hilfe des Refresh-Headerfelds eine Art Diashow einrichten. Sie brauchen dazu nur die einzelnen Webseiten mit Ihren Photos aufzusetzen und jede Seite mit einem Refresh-Headerfeld zu versehen, der zur nächsten Seite in der Diashow weiterleitet.



Neben dem Refresh-Headerfeld kann man auch JavaScript zur automatischen Weiterleitung verwenden (siehe Kapitel 7).

Seitenübergangseffekte (Internet Explorer)

Natürlich gibt es auch Headerfelder, die von dem einen oder anderen Browser definiert und unterstützt werden, während sich alle anderen Browser weigern, diese Headerfelder anzuerkennen (oder in der Implementierung einfach etwas hinterher hinken). Hierzu gehören beispielsweise die Seitenübergangseffekte des Internet Explorers.

Was geschieht üblicherweise, wenn der Anwender auf einen Hyperlink klickt, der zu einer anderen Webseite führt? Der Browser löscht die aktuelle Webseite aus seinem Fenster und baut dann nach und nach die neue Webseite auf.

Ein Seitenübergang ist dagegen ein Überblendeffekt, der festlegt, wie die aktuelle Webseite Schritt für Schritt durch die neue Webseite zu ersetzen ist.



Da Seitenübergänge als simulierte HTTP-Headerfelder realisiert werden, werden sie von Browsern, die mit diesen Headerfeldern nichts anfangen können, einfach ignoriert. Sie können also ruhig für alle Freunde des Internet Explorers Seitenübergänge einbauen, ohne befürchten zu müssen, dass Navigator-Fans beim Anschauen Ihrer Webseiten mit Fehlermeldungen überschüttet werden.

Seitenübergänge werden in drei Schritten erstellt:

1. Zuerst legt man fest, bei welchem Ereignis der Überblendeffekt ausgeführt werden soll. Den Namen des Ereignisses geben Sie als Wert für das http-equiv-Attribut an.

```
<meta http-equiv="page-exit"
```

Folgende Ereignisse sind definiert:

Attribut-Wert	Beschreibung
page-enter	Die Webseite wird betreten. Unserer Erfahrung nach tritt der Effekt nur ein, wenn man die Webseite über den Zurück-Schalter des Browsers ansteuert.
page-exit	Die Webseite wird verlassen.
site-enter	Die Website wird betreten. Unserer Erfahrung nach tritt der Effekt nur ein, wenn man die Webseite über den Zurück-Schalter des Browsers ansteuert. Hinweis: Um den Effekt auszuprobieren, laden Sie die Webseite über ihren URL von localhost, rufen Sie dann eine Webseite einer andere Internet-Website oder eine Datei auf Ihrer Festplatte auf, und kehren Sie dann wieder zurück.
site-exit	Die Website wird verlassen.

2. Danach rufen Sie die Übergangsfunktion auf und legen fest, wie lange der Übergang dauern soll.

Es gibt zwei Übergangsfunktionen: `blendTrans` und `revealTrans`. Die Funktion `blendTrans` erzeugt einen Fading-Effekt, was ungefähr so aussieht, wie wenn in einem Film ein Geist durch eine Wand tritt. Meines Erachtens einer der schönsten Übergangseffekte.

Als ersten Parameter übergeben Sie der Übergangsfunktion eine Zeitangabe, die festlegt, wie viele Sekunden der Effekt andauern soll. Beachten Sie aber, dass der Effekt bei der Ausführung im Browser meist mehr Zeit beansprucht als Sie im `<meta>`- Tag vorgeben. Wählen Sie daher eher etwas kürzere Zeiten (beispielsweise zwischen 0.5 und 5).

```
<meta http-equiv="page-exit"
      content="blendTrans(Duration=2.0)" />
```



Abbildung 6.6: Überblendung zwischen kiefer.html und seite_a.html

Wenn Sie sich für `blendTrans` entschieden haben, sind Sie damit schon fertig. Wenn Sie sich für `revealTrans` entschieden haben, müssen Sie danach noch angeben, welchen Effekt genau Sie verwenden möchten.

3. Für `revealTrans` müssen Sie als zweiten Parameter noch einen Zahlencode für den gewünschten Effekt angeben:

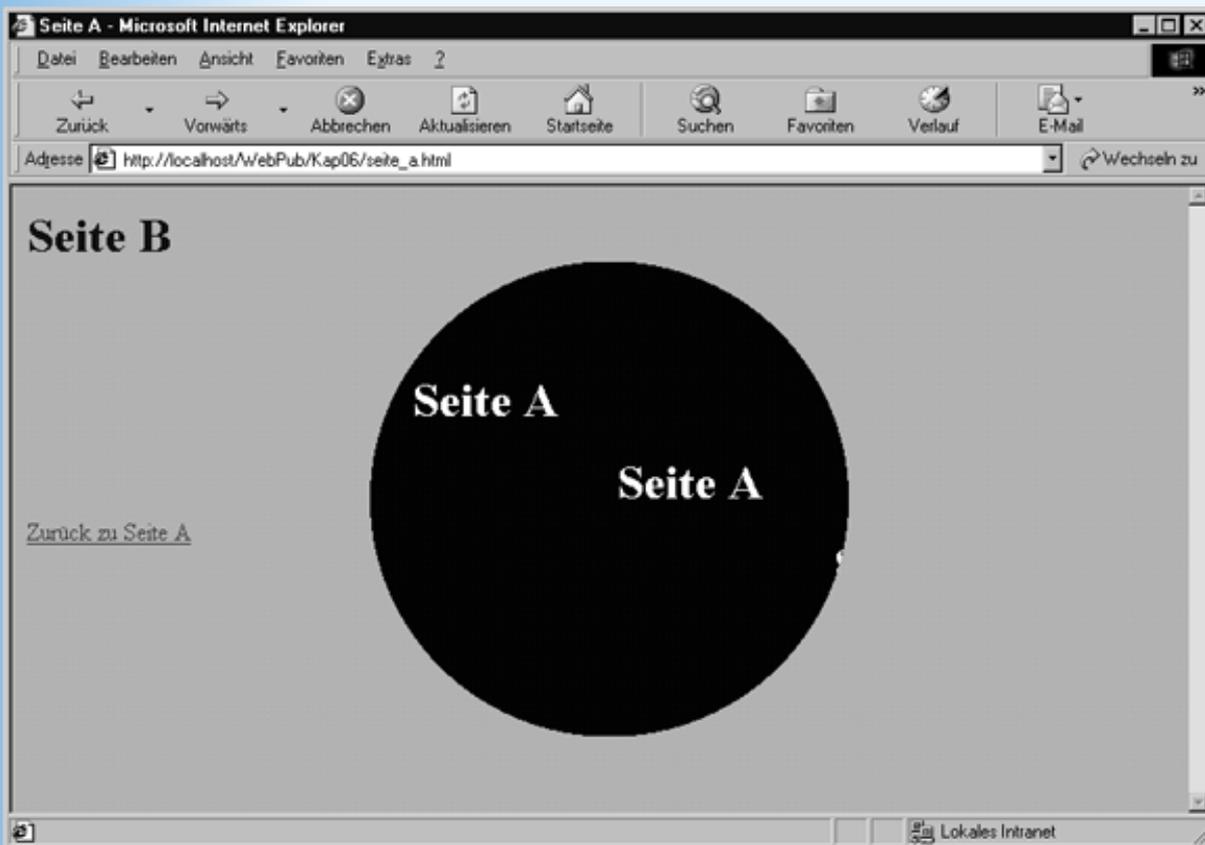
```
<meta http-equiv="page-exit"
      content="revealTrans(Duration=2.0,Transition=2)" />
```

Folgende Effekte stehen zur Verfügung:

Code	Effekt
0	Von außen einblenden Die alte Webseite verschwindet in einem immer kleiner werdenden Rechteck.
1	Von innen einblenden Die neue Webseite erscheint in einem immer größer werdenden Rechteck.
2	Spirale nach innen Die alte Webseite verschwindet in einem immer kleiner werdenden Kreis.

3 Spirale nach außen

Die neue Webseite erscheint in einem immer größer werdenden Kreis.



4 Von unten rollen

Die neue Webseite wird zeilenweise von unten aufgebaut. Der Effekt sieht ungefähr so aus, als würde ein Vorhang hochgezogen.

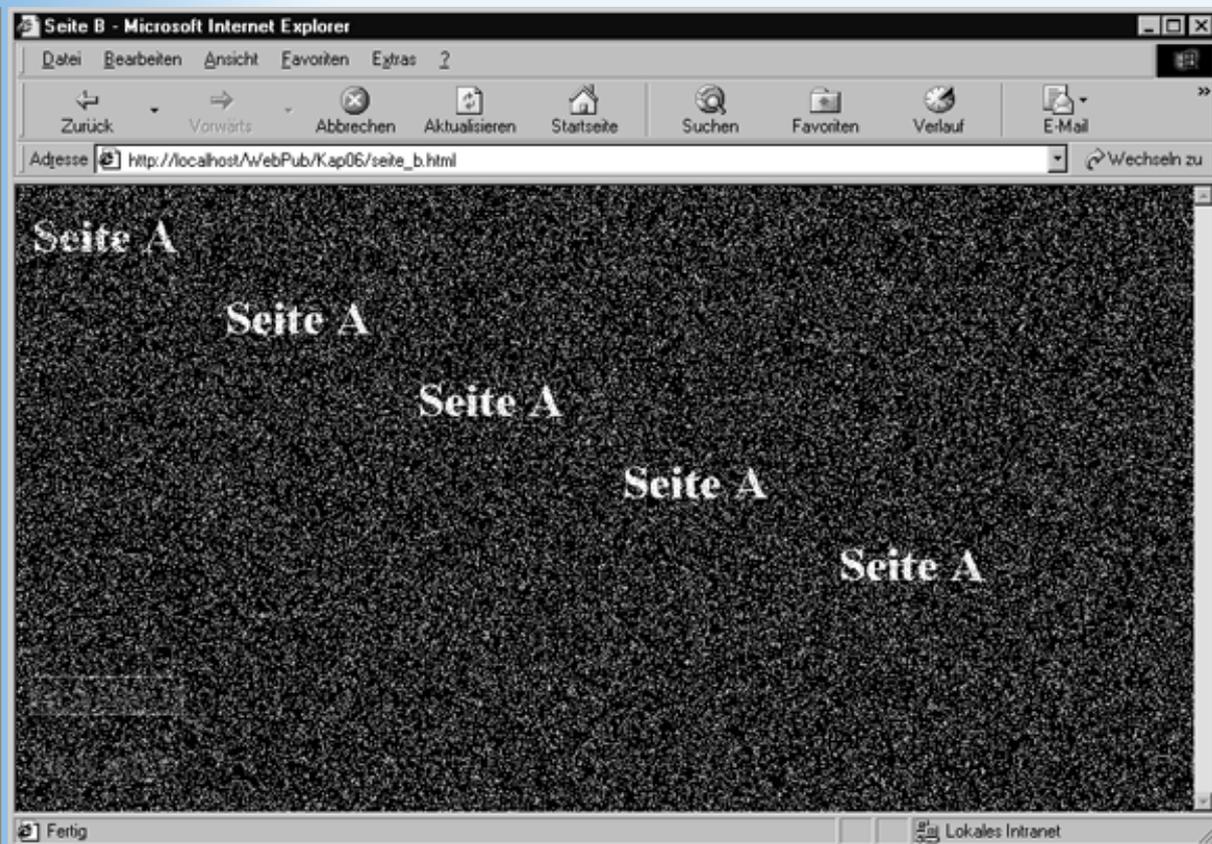
Für den zeilenweisen Aufbau von oben, links und rechts gibt es die Effekte 5, 6 und 7.

8 Vertikale Jalousie

Ein Effekt, der an das Schließen (oder Öffnen) einer Jalousie mit senkrechten Lamellen erinnert. Die Codenummern 9, 10 und 11 stehen für weitere Jalousie-Effekte.

12 Auflösen

Einer der besten Effekte! Die Webseite löst sich pixelweise auf.



13	Vertikal schließen Gleicht den Rolleeffekten, nur dass die Webseite von beiden Seiten ausgeblendet wird.
14	Horizontal schließen Die Webseite wird von oben und unten ausgeblendet.
17	Diagonaler Rolleeffekt, bei dem die alte Webseite von rechts oben nach links unten verschwindet. Die Codenummern 18, 19 und 20 stehen für weitere diagonale Rolleeffekte.
21	Horizontale Linien Die neue Webseite wird Zeile für Zeile eingeblendet. Die Zeilen werden aber nicht von oben nach unten, sondern in zufälliger Reihenfolge eingeblendet.
22	Vertikale Linien Die neue Webseite wird Zeile für Zeile eingeblendet. Die Zeilen werden aber nicht von links nach rechts, sondern in zufälliger Reihenfolge eingeblendet.
23	Zufallsgenerator Der Überblendeffekt wird erst bei Eintritt des Ereignisses ausgewählt.

Tabelle 6.5: Seitenübergangseffekte des Internet Explorers

Auf der Buch-CD finden Sie zwei Webseiten *seite_a.html* und *seite_b.html* mit denen Sie die Effekte ausprobieren können. Mit dem *file*-Link auf der Webseite *seite_a.html* können Sie das Verlassen der Website testen. Die Pfadangabe des Links müssen Sie gegebenenfalls anpassen.



Grundsätzlich sollten Sie Seitenübergänge nur sparsam einsetzen und nicht zu lange andauern lassen, sonst werden die Websurfer Ihre Seiten irgendwann enerviert meiden. Lassen Sie die Effekte aber auch nicht zu schnell ablaufen, sonst verlieren Sie unter Umständen ihre Wirkung. Hier ist Fingerspitzengefühl und viel Testarbeit gefragt.

6.4 Suchmaschinen unterstützen

Zu guter Letzt müssen Sie Ihre Website im World Wide Web bekannt machen, denn was nützt die schönste Website, wenn sie von niemandem gefunden wird. Schauen wir uns also an, wie Sie Ihre Webseite in vier Schritten vor einem Dornröschenschlaf bewahren können.

Unterstützen Sie Suchmaschinen und Web Crawlers

Stellen Sie sich vor, Sie hätten ein Web mit Gedichten ins Internet gestellt. Neben eigenen Werken finden sich in der Sammlung auch Werke von Rilke, Longfellow und einigen chinesischen Dichtern, die Sie besonders mögen, und Sie möchten nun dafür sorgen, dass Websurfer, die mit Hilfe gängiger Suchmaschinen (Lycos, Fireball, Yahoo, etc.) nach Gedichten oder den Namen der betreffenden Dichter suchen, auch auf Ihre Webseiten aufmerksam werden.

Üblicherweise bestehen die großen Suchmaschinen aus drei Teilen:

- einer indizierten Datenbank, in der ein Großteil der im Internet veröffentlichten Webseiten erfasst ist
- der eigentlichen Suchmaschine, in die der Websurfer seine Suchbegriffe eingibt und die daraufhin in der Datenbank nachschaut, welche Webseiten zu den Suchbegriffen passen
- einem sogenannten Webcrawler oder Webroboter, der sich von Link zu Link durch die Webseiten des Internets hangelt und die gefundenen Webseiten gemäß ihren Schlüsselwörtern katalogisiert und in die Datenbank der Suchmaschine einträgt.

Suchmaschinen

Suchmaschinenbetreiber sind kommerzielle Unternehmen. Sie leben nicht von den Leuten, die ihre Suchmaschine verwenden, sondern von den Werbeeinnahmen, die sie kassieren, wenn viele Internetnutzer gerade ihre Suchmaschine auswählen.

Die Betreiber haben sich deshalb ein Rankingsystem ausgedacht, wie sie die in Ihrer Datenbank abgelegten Informationen einer HTML-Seite interpretieren, um sie für die Benutzer attraktiv zu halten. Der Nutzer des Suchdienstes möchte ja genau die Information finden, die er erwartet, und nicht auf Seiten geführt werden, die mit seinem Suchbegriff nichts zu tun haben, nur weil ein findiger Programmierer das Rankingsystem des Betreibers ausgetrickst hat (weshalb die Suchmaschinenbetreiber Ihre Strategie auch des öfteren ändern). Neben dem Domainnamen werden das TITLE-Tag und die Schlüsselwörter im META-Tag (siehe unten) in die Bewertung einbezogen (mit unterschiedlicher Gewichtung). Es wird aber auch geprüft, ob diese Stichwörter als Text in der Seite wieder vorkommen. Daraus resultiert dann eine höhere Wertigkeit und ein höherer Stellenwert bei der Anzeige der Suchergebnisse. Die neuesten Gewichtungsfunktionen beziehen sogar die Publizität einer Seite mit in die Wertung ein. Dabei wird nachgeschaut, wie oft Links von anderen Webseiten auf die gesuchte Webseite verweisen, denn dann muss die gesuchte Seite ja eine gewisse Attraktivität haben.

Mit Hilfe einiger simpler Tricks können Sie Webcrawlern, die auf Ihre Seiten stoßen, bei der Auswertung und Katalogisierung behilflich sein.

- Geben Sie die Schlüsselwörter an, unter denen Ihre Webseite in der Datenbank eingetragen werden soll. (Dies sind die Stichwörter, die Websurfer als Suchbegriffe in die Suchmaschinen eingeben.)

Setzen Sie hierzu im Header-Abschnitt ein META-Element mit dem Namen »keywords« auf und geben Sie als content die Liste der Schlüsselwörter an:

```
<meta name="keywords" content="Gedichte,Rilke,Wang Wei,Longfellow">
```



Versuchen Sie nicht, mehr Besucher auf Ihre Website zu ziehen, indem Sie Schlüsselwörter angeben, die sehr attraktiv sind (beispielsweise »Geld«), aber nichts mit dem Inhalt Ihrer Webseiten zu tun haben. Erstens prüfen die meisten Webcrawler mittlerweile, ob sich die genannten Schlüsselwörter auch im Text der Seiten wiederfinden, zweitens kommen solche Mogeleyen bei den Websurfern nicht gut an und drittens ist der Erfolg solcher Tricks fraglich, wenn die eigene Webseite dann als 1054-ter Eintrag zum Stichwort »Geld« aufgeführt wird.

- Setzen Sie eine kurze Beschreibung auf, die auf den Trefferseiten der Suchmaschinen neben oder unter der Webadresse Ihrer Webseite angezeigt wird.

```
<meta name="description" content="Gedichtsammlung mit eigenen Werken und Gedichten ausgewählter Dichter (Schwerpunkt auf alten chinesischen Dichtern) " />
```

- Zeigen Sie dem Webcrawler an, in welcher Sprache Ihre Webseite verfasst ist.

Sie brauchen dazu nur im öffnenden <html>-Tag das lang-Attribut zu definieren und diesem den entsprechenden Sprachcode zuzuweisen:

```
<html lang=de>
```



Die wichtigsten Sprachcodes sind en (Englisch), en-US (Amerikanisch), fr (Französisch), de (Deutsch), it (Italienisch), nl (Holländisch), el (Griechisch), es (Spanisch), pt (Portugiesisch), ar (Arabisch), he (Hebräisch), ru (Russisch), zh (Chinesisch), ja (Japanisch), hi (Hindi), ur (Urdu) und sa (Sanskrit).

- Wenn Sie Ihr Web nicht nur auf Deutsch, sondern auch in anderen Sprachen anbieten, sollten Sie es nicht versäumen, Webcrawler auf die anderen Sprachseiten hinzuweisen.

Wenn Sie Ihre Webseiten neben Deutsch auch in Englisch und Spanisch anbieten, sollten Sie auf der deutschen Startseite unbedingt im Header-Abschnitt <link>- Verweise auf die englische und die spanische Startseite angeben, damit englisch/ amerikanische und spanische Webcrawler direkt den Weg zu diesen Seiten finden. Verwenden Sie dazu das <link>-Tag mit der rel-Beziehung alternate und geben Sie die Zielsprache über das Attribut hreflang an.

```
<link rel="alternate"
      type="text/html"
      href="start-en.html" hreflang="en"
      lang="en" title="Poems" />
```

```
<link rel="alternate"
      type="text/html"
      href="start-es.html" hreflang="es"
      lang="es" title="Antolog&iacute;a en verso" />
```



Das lang-Attribut im link-Tag bezieht sich eigentlich nur auf die Sprache, in der der Wert des title-Attributs verfasst ist (wichtig, wenn in diesem Sonderzeichen wie Umlaute oder Buchstaben mit Akzenten auftauchen), es kann aber nicht schaden, es zu setzen.

- Zeigen Sie dem Webcrawler die Startseite Ihrer Gedichtesammlung an.

Setzen Sie dazu im Header-Abschnitt ein weiteres LINK-Element mit der rel- Beziehung start auf.

```
<link rel="start"          type="text/html"
      href="page1.html" title="Meine Gedichtesammlung" />
```

- Zeigen Sie dem Webcrawler in welchen Abständen er Ihre Webseite auf Aktualisierungen überprüfen soll.

```
<meta name="revisit-after" content="30 days" />
```



Schließlich gibt es auch die Möglichkeit, eine Webseite von der Indexierung durch Webcrawler auszuschließen. Der beste Weg hierfür besteht darin, im Website-Verzeichnis eine Datei robots.txt aufzusetzen und in dieser die auszuschließenden Webseiten aufzuführen (siehe beispielsweise <http://www.w3.org/robots.txt>). Die Datei robots.txt wird allerdings nur im obersten Verzeichnis einer Website (Webdomäne) beachtet, nicht in den Unterverzeichnissen der Webs (wenn die Webs mehrerer Mitglieder unter einer Domäne abgelegt sind). Wenn Sie Ihr Web also beispielsweise in einem eigenen Unterverzeichnis ablegen, müssen Sie sich zur Anpassung von robots.txt an Ihren Internet-Provider wenden. Alternativ können Sie Ihre Webseiten um ein <meta>-Element mit dem Namen robots erweitern und als content einen der folgenden Werte angeben: noindex (nicht indexieren), nofollow (nicht den enthaltenen Links folgen), index (indexieren und in die Suchmaschine aufnehmen), all (indexieren und Links folgen).

Melden Sie sich bei verschiedenen Suchmaschinen an

Bei manchen Suchmaschinen kann (muss) man sich direkt anmelden, um in den Katalog aufgenommen zu werden.

<http://www.yellow.com>

<http://www.yahoo.com>

<http://www.submit-it.com>



Bei manchen Suchmaschinen ist der direkte Eintrag in den Katalog mit einer Gebühr verbunden.

Verschicken Sie E-Mails.

Schicken Sie E-Mails an Ihre Freunde, Bekannte und Kollegen und werben Sie für Ihre Website.

Setzen Sie sich mit anderen Webautoren zusammen

Setzen Sie sich mit anderen Webautoren in Verbindung, die über die gleichen Interessengebiete publizieren und fragen Sie an, ob diese Links auf Ihre Homepage einrichten möchten.

6.5 Zusammenfassung

Am heutigen Tag haben wir drei Themengebiete behandelt: die Einbindung von Multimedia-Dateien, die Anzeige von Quickinfos und die Tags für den <head>-Abschnitt.

Traditionell werden Multimedia-Dateien mit Hilfe des <embed>-Tags in Webseiten integriert (alternativ kann man Hyperlinks auf die Dateien anbieten). Das <embed>-Tag wurde allerdings nie in den HTML-Standard aufgenommen. Dieser spezifiziert stattdessen das <object>-Tag, mit dem man neben Multimedia-Dateien auch Programmobjekte (beispielsweise Applets oder ActiveX-Steuer-elemente) einbetten kann. Leider wird das <object>-Tag von den gängigen Browsern in ganz unterschiedlicher Art und Weise unterstützt und eignet sich derzeit eigentlich vornehmlich für die Integration von Programmobjekten (siehe Kapitel 15 und 16).

Relativ einfach gestaltet sich die Einrichtung von Quickinfos - kleine (Hilfe-)Texte zu HTML-Elementen die eingeblendet werden, wenn die Maus über dem zugehörigen Element verweilt. Für Quickinfos muss man nur das title-Attribut setzen (und hoffen, dass die Browser der Webbesucher Quickinfos anzeigen).

Schließlich haben wir uns die Tags für den <head>-Abschnitt angeschaut: <base>, <link> und <meta>, wobei wir uns besonders ausführlich mit den Seitenübergängen des Internet Explorers und der browserunabhängigen - Unterstützung von Suchmaschinen beschäftigt haben.

6.6 Fragen und Antworten

Frage:

Warum wird das <object>-Tag derzeit so unvollständig unterstützt?

Antwort:

Zum einen weil die Webdesigner seit Jahren das <embed>-Tag zur Einbindung von Multimedia-Inhalten verwenden (folglich sind die Browser-Hersteller quasi gezwungen das <embed>-Tag weiter zu unterstützen. Zum anderen weil Microsoft schon vor dem HTML-Standard ein <object>-Tag definiert hat, dass allerdings lediglich zur Einbindung von Programmobjekten (insbesondere ActiveX- Steuer-elementen) vorgesehen war.

Frage:

Wozu soll ich Seitenübergänge einrichten, die man nur im Internet Explorer sieht?

Antwort:

Seitenübergänge können, sparsam eingesetzt, ganz nett sein, und der Internet Explorer wird derzeit immerhin von rund 60% aller Websurfer benutzt. Außerdem entsteht kein Schaden, wenn Websurfer Browser verwenden, die die Seitenübergänge nicht unterstützen.

Frage:

Es gibt so viele META-Tags. Muss ich diese Tags alle verwenden?

Antwort:

Nein, alle META-Tags sind optional. Es empfiehlt sich allerdings, von den META-Tags zur Angabe der Medieninhalte (welcher Stylesheet-Typ), zur Unterstützung der Suchroboter und zur Angabe des Autors eifrig Gebrauch zu machen (es ist ja schließlich nur ein wenig Tipparbeit).

Frage:

Ich habe von Diensten gehört, die anbieten, mich im Suchmaschinenranking an die vorderen Positionen zu bringen. Ist das seriös?

Antwort:

Je nach Wichtigkeit einer Seite kann es sinnvoll sein, das Ranking einer Seite genauestens zu verfolgen und gegebenenfalls zu fördern. Dafür gibt es Dienste, die aber nur funktionieren können, wenn sie entsprechend teuer sind, damit sich nicht jeder diesen Service leisten kann. Es kann ja immer nur ein Eintrag an erster Stelle in der Suchmaschine erscheinen.

Frage:

Freunde empfehlen mir darüber hinaus, möglichst viele Schlüsselwörter anzugeben, um mich an die Spitze der Suchergebnisse zu bringen. Was ist davon zu halten?

Antwort:

Die Suchmaschinen beschränken die Anzahl der Suchbegriffe, indem Sie die maximale Zahl von Zeichen für alle Suchbegriffe einer Webseite auf 512 bis 1024 Buchstaben festlegen. Mehr Zeichen werden nicht eingelesen, so dass eine endlose Aneinanderreihung von Suchbegriffen sinnlos ist. Die Begriffe sollten natürlich auch etwas mit der auf Ihren Seiten zu findenden Informationen zu tun haben, sonst verlässt ein Internetsurfer ihre Seite genervt sehr schnell wieder.

Es nutzt auch nichts, einen Begriff bei den Schlüsselworten möglichst oft zu wiederholen. Die Suchmaschinen erkennen das sofort und werten Ihre Seite ab oder nehmen Sie gar nicht erst in die Datenbank auf.

6.7 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

Quiz

1. Wie kann man auf einer Webseite Sounddateien zum Abspielen anbieten?
2. Warum sollten Quickinfos keine wichtigen Informationen enthalten?
3. Eine Webseite spezifiziert folgende Basisadresse: `http://www.meinServer.de/Verz.` welche absoluten URLs erzeugt der Browser dann für folgende relativen URLs?
 - `demo1.html`
 - `./bilder/bild.gif`
 - `../cgi-bin/prog.pl`
4. Was ist ein Webcrawler oder Suchroboter?
5. Welche beiden META-Tags sollte man mindestens zur Unterstützung von Suchrobotern aufsetzen?

Übungen

1. Setzen Sie eine Webseite mit Hyperlinks zum Abspielen von verschiedenen WAV- Dateien auf.
2. Erstellen Sie eine Eingangsseite zu einem Web (farbiger Hintergrund, kurzer Willkommenstext, Hyperlink zur Hauptseite des Webs). Versehen Sie die Webseite mit META-Tags zur Unterstützung von Webcrawlern und richten Sie einen unsichtbaren Textabschnitt ein, in dem Sie den Inhalt des Webs noch einmal beschreiben und alle Schlüsselwörter aus dem Header-Abschnitt verwenden.³

1

Beachten Sie, dass einige WebspacerProvider die beschriebene »Umleitung« in ihren Verträgen ausschließen, da sonst ja der Traffic für zwei Webserver zu ihren Lasten geht.

2

Dies gilt nicht für Grafikdateien. Wenn man also eine Grafikdatei überarbeitet und deren Namen beibehält, kann es lange dauern, bis diese neuen Informationen überall zur Verfügung stehen.

3

Es gibt allerdings keine Garantie dafür, dass der Webcrawler den verborgenen Text akzeptiert.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

Tag 7

Ein gelungener Webauftritt

Puuuh, die Woche war anstrengend. Dafür werden wir die Woche ein wenig gemächlicher ausklingen lassen. Wir werden auf der Basis der bisher gelernten Techniken ein eigenes kleines Web erstellen und uns ein wenig über Webseiten und Design im Allgemeinen unterhalten.

Die Themen heute:

- Erstellung eines Webs auf der Basis einer Frameseite
- Einheitliches Design mit Stylesheets
- Unterstützung von Suchmaschinen
- Einrichtung einer Eingangsseite
- Beispiele für interessante Websites
- Grundregeln des Webdesigns

7.1 Mit einfachem Design anfangen

Wer in der Erstellung von Webseiten noch nicht so erfahren ist, sollte nicht der Versuchung erliegen, gleich im ersten Anlauf »die welterschütternde Website« kreieren zu wollen. Wichtiger ist es, sich erst einmal ein klares Konzept für den Aufbau des Webs und die Navigation durch die Webseiten zu erarbeiten. Die Profis sprechen in diesem Zusammenhang auch von der »Usability« einer Website.



Usability ist ein Maß für die Bedienbarkeit und Benutzerfreundlichkeit eines Systems (beispielsweise eines Webs). Kriterien für die Usability eines Systems sind: Lernkurve, Effizienz der Bedienung, Statusverwaltung, Fehlerbewältigung und subjektives Gefühl (fühlt sich der Anwender wohl beim Umgang mit dem System).

Für die Usability einer Webseite ist vor allem eine steile Lernkurve wichtig. Der Websurfer will nicht erst umständliche Bedienungsanleitung lesen müssen, um sich im Web hin und her zu bewegen oder mit den Angeboten im Web zu interagieren. Hier kann man durch logische Organisation des Webs und intuitiv nachvollziehbare Navigationsstrukturen viel bewirken.



Umgekehrt können gerade ungewöhnliche Design- und Navigationslösungen das Interesse des Websurfers wecken und ihn anregen, die Website zu erforschen (siehe beispielsweise

Website der Mailänder Scala in Abbildung 7.7). Welchen Weg man einschlägt hängt letztlich also vor allem von der Art der Website und dem Profil der anzusprechenden Websurfer ab.

Organisation vor Design (frame_seite.html)

Im Falle unseres kleinen Beispielwebs wollen wir unsere Design-Ansprüche der Organisation und Navigierbarkeit des Webs unterordnen. Konkret soll das heißen, dass wir die Homepage des Webs als Frameseite aufsetzen werden und die dadurch bedingten Einschränkungen in der Wahl des Designs in Kauf nehmen.

Die Aufteilung in Frames (Banner, Inhaltsverzeichnis, Anzeigebereich) hat für uns den Vorteil,

- dass das Web von Anfang an über eine klare Organisation und eine den Websurfern wohl vertraute Navigationsstruktur verfügt
- dass wir mit der Implementierung der Navigationsstruktur wenig Arbeit haben (viele nimmt uns die Frame-Technik ab)
- dass man später bei Bedarf ohne allzu große Mühen auf ein frameloses Design umstellen kann.

Die Frames der Homepage

Unser Beispielweb ist eine kleine Gedichtesammlung, die wir ins Internet stellen wollen. Abbildung 7.1 vermittelt Ihnen einen Eindruck davon, wie unser Web später aussehen wird.

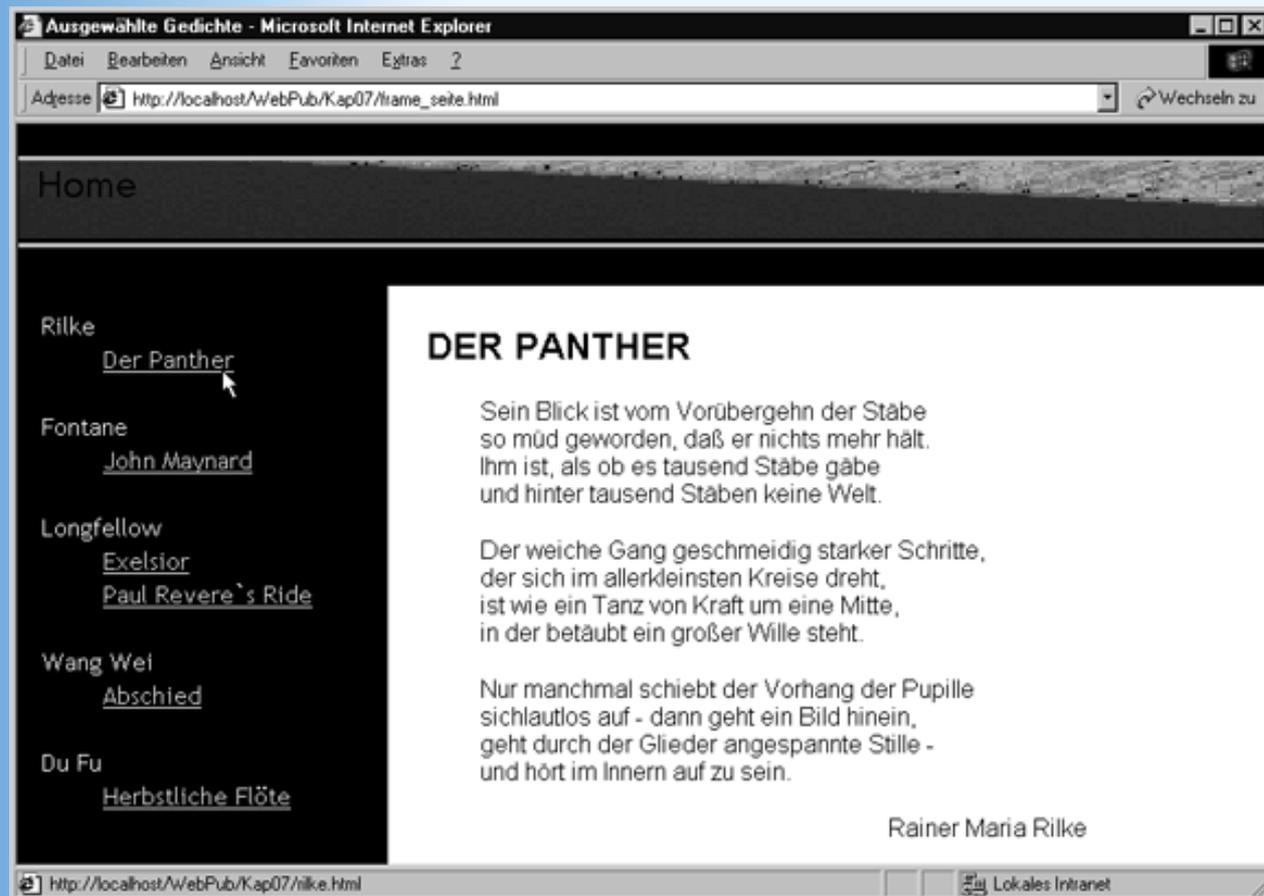


Abbildung 7.1: Die Startseite des Webs (das Design hat durch die Umwandlung in Grauwerte etwas gelitten)

Die Frame-Definition für diese Webseite sieht wie folgt aus:

```
<frameset rows="105,*" border="0">
```

```

<frame name="oben" scrolling="no" noresize src="oben.html"
      frameborder="0"
      marginwidth="0" marginheight="0">
<frameset cols="240,*">
  <frame name="inhalt" target="hauptframe" src="inhalt.html"
        frameborder="0">
  <frame name="hauptframe" src="begruessung.html" frameborder="0" >
</frameset>

```

Zuerst werden zwei Zeilen definiert, von denen die erste 105 Pixel hoch ist. In ihr werden wir ein grafisches Banner einblenden. Die zweite Zeile, die den Rest des Browserfensters einnimmt, ist in zwei Spalten aufgeteilt. Im ersten (linken) Frame blenden wir das Inhaltsverzeichnis unseres Webs ein (eine Sammlung von Links zu den Gedichten), der rechte Frame dient der Anzeige der Gedichte (zu Beginn geben wir in diesem Frame einen kleinen Begrüßungstext aus).

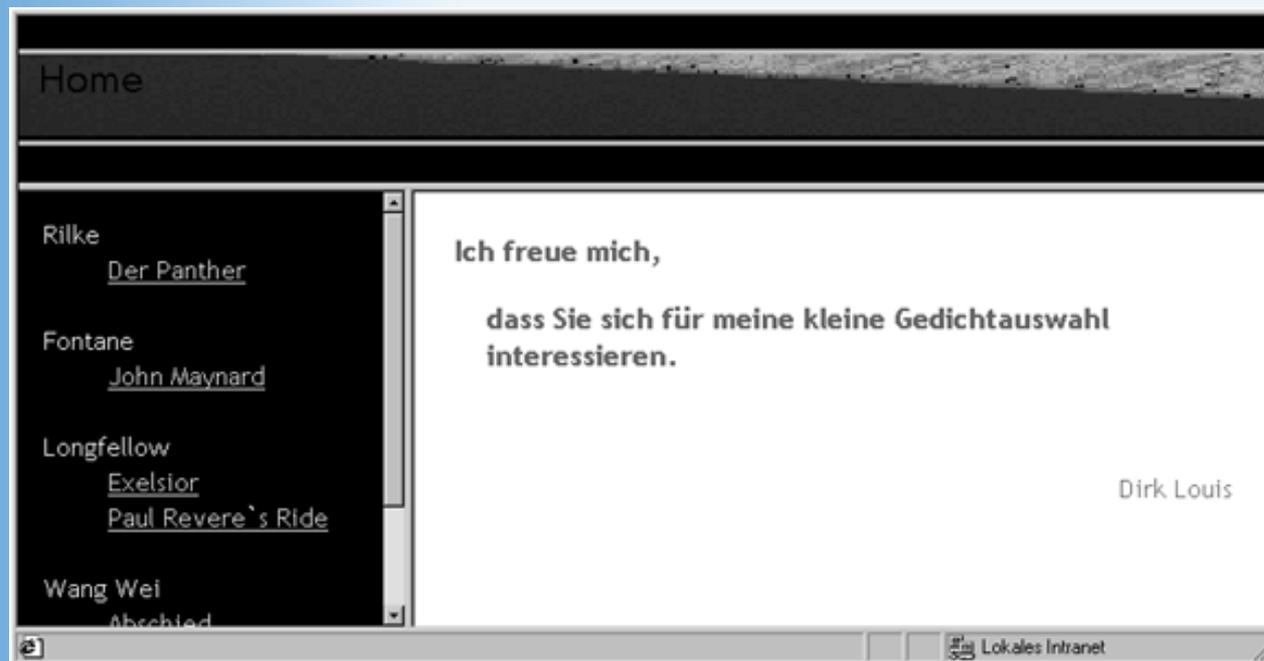


Abbildung 7.2: Die Aufteilung in Frames

Im endgültigen Web sind die Frames ausgeblendet. Dazu werden die `frameborder`-Attribute der Frames auf 0 gesetzt. Das `border`-Attribut des `Framesets` sorgt dafür, dass keine Leerräume zwischen den Frames zurückbleiben.

Für den oberen Frame setzen wir auch noch die Attribute `marginwidth` und `marginheight` auf 0. So können wir sicherstellen, dass unser Banner-Bild in allen Browsern direkt am Rand des Browserfensters beginnt (speziell der Navigator 4 tendiert ansonsten dazu alle Frameinhalte einzurücken).

Der wichtigste Teil der Frameseite ist damit schon fertig. Jetzt müssen wir nur noch META-Tags für die Unterstützung von Suchrobotern einfügen (unsere Website soll ja schließlich auch besucht werden) und eine alternative Darstellung für Browser ohne Frame-Unterstützung vorsehen.

Listing 7.1: frame_seite.html

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>

```

```

<title>Ausgewählte Gedichte</title>
<!-- Suchmaschinen unterstützen -->
<meta name="language" content="de">
<meta lang="de" name="keywords" content="Gedichte,Rilke,
    -Wang Wei,Longfellow, Dufu, Fontane">
<meta name="description" content="Gedichtsammlung mit eigenen Werken
    -und Gedichten ausgewählter Dichter (Schwerpunkt auf alten
    -chinesischen Dichtern) ">
<meta name="revisit-after" content="30 days">
<meta name="robots" content="all">
</head>
<frameset rows="105,*" border="0">
  <frame name="oben" scrolling="no" noresize src="oben.html"
    frameborder="0"
    marginwidth="0" marginheight="0" ><!-- kein Standardrand durch
        Browser -->
  <frameset cols="240,*">
    <frame name="inhalt" target="hauptframe" src="inhalt.html"
      frameborder="0">
    <frame name="hauptframe" src="begruessung.html" frameborder="0" >
  </frameset>
</frameset>
<noframes>
  <body>
    <p>Diese Frame-Seite enthält eine Gedichtesammlung mit ausgewählten
    Gedichten von Rilke, Wang Wei, Longfellow, Dufu und Fontane. Leider unterstützt
    Ihr Browser keine Frames. Wechseln Sie zur Seite <a
    href="alternativ.html">OHNE_FRAMES</a>. </p>
  </body>
</noframes>
</frameset>
</html>

```

Die META-Tags zur Unterstützung der Suchmaschinen haben wir in Kapitel 6.4 besprochen. Den alternativen <body>-Teil nutzen wir dazu, die Stichworte aus den META- Tags im Seiteninhalt zu wiederholen. (Denken Sie daran, dass Suchroboter üblicherweise nicht den Links der Frames folgen.)

Das Banner (oben.html)

Die Seite oben.html enthält nicht mehr als ein einfaches Banner (eine langgestreckte Grafik) mit einem einzelnen Link, über den der Webbesucher auf die Begrüßungsseite zurückkehren kann, nachdem er zuvor das eine oder andere Gedicht gelesen hat.

Listing 7.2: oben.html

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Banner</title>
  <!-- Stylesheets -->
  <meta http-equiv="Content-Style-Type" content="text/css">
  <link rel="StyleSheet" type="text/css" href="gedichte_ingang.css">
</head>

```

```

<body>
<div class="zentriert">
  
  <map name="map1">
    <area href="begrueessung.html" target="hauptframe" shape="rect"
      coords="3,3,90,33"></area>
  </map>
</div>
</body>
</html>

```

Zur Formatierung verwendet die Seite ein externes Stylesheet.

Listing 7.3: gedichte_eingang.css

```

body      { background-color: black; color: white;
            margin-top: 20px; margin-left: 0px;
            font-family: 'Trebuchet MS', Arial, Helvetica, sans-serif;
            font-size: 12pt }
p.dichter { margin-left:300px }
.zentriert { text-align: center }
#zaehler  { text-align: right; font-family: sans-serif; font-size: 8pt}
#verborgen { visibility: hidden; color: black }

```

Die Seite oben.html verwendet nur die Stylesheets body und .zentriert. Die anderen Stylesheets brauchen wir für die Eingangsseite, die wir weiter unten erstellen werden und die die gleiche Stylesheet-Datei einbindet.

Das Inhaltsverzeichnis (inhalt.html)

Das Inhaltsverzeichnis besteht im Wesentlichen aus einer einzigen Definitionsliste. Die »zu definierenden Begriffe« sind die Dichter, die »Definitionen« sind die Gedichte.

Listing 7.4: inhalt.html

```

<html>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Inhaltsverzeichnis</title>
  <!-- Stylesheets -->
  <meta http-equiv="Content-Style-Type" content="text/css">
  <link rel="StyleSheet" type="text/css" href="gedichte_inhalt.css">
</head>
<body>
<dl>
  <dt>Rilke</dt>
  <dd><a href="rilke.html" target="hauptframe">Der Panther</a></dd>
  <dt>&nbsp;</dt>
  <dt>Fontane</dt>
  <dd><a href="fontane.html" target="hauptframe">John Maynard</a></dd>
  <dt>&nbsp;</dt>

```

```

<dt>Longfellow</dt>
<dd><a href="longfellow.html#excelsior"
      target="hauptframe">Exelsior</a><br />
      <a href="longfellow.html#paulrevere"
      target="hauptframe">Paul Revere`s Ride</a></dd>
<dt>&nbsp;</dt>
<dt>Wang Wei</dt>
<dd><a href="wangwei.html" target="hauptframe">Abschied</a></dd>
<dt>&nbsp;</dt>
<dt>Du Fu</dt>
<dd><a href="dufu.html" target="hauptframe">Herbstliche Flöte</a></dd>
</dl>
</body>
</html>

```

Die Datei *inhalt.html* verwendet zur Formatierung ihre eigene externe Stylesheet-Datei.

Listing 7.5: gedichte_inhalt.css

```

body      { background-color: black; color: white;
           margin-left: 15px;
           font-family: 'Trebuchet MS', Arial, Helvetica, sans-serif }
a:link    { color: #2071CC }
a:visited { color: #2071CC }
a:active  { color: red }

```

Mit Hilfe der Stylesheets legen wir neben der zu verwendenden Schriftart und dem Abstand zum linken Rand vor allem die Farben fest. Neben Hintergrund- und Vordergrundfarbe definieren wir auch zum Hintergrund passende Hyperlinkfarben.

Die Webseiten für die Gedichte

Die Hyperlinks aus der Inhaltsseite laden die Webseiten der zugehörigen Dichter in den Hauptframe (und springen gegebenenfalls zu dem gewünschten Gedicht, wenn es zu einem Dichter mehrere Gedichte gibt).

Die Webseiten zu den Dichtern sind immer gleich aufgebaut und verwenden alle das gleiche externe Stylesheet.

Listing 7.6: wangwei.html - Beispiel für die Webseite zu einem Dichter

```

<html>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
      "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>ABSCHIED</title>
  <!-- Stylesheets -->
  <meta http-equiv="Content-Style-Type" content="text/css">
  <link rel="StyleSheet" type="text/css" href="gedichte.css">
</head>
<body>
<h2>ABSCHIED</h2>
<div class="gedicht">

```

```

<p>Steig doch vom Pferd, komm, laß uns einen Becher leeren,<br />
Verrate mir, wohin die Reise geht!<br />
Du sagst, dir sei im Leben nichts gelungen,<br />
Nun kehrst du heim, am Rand des Südgebirges auszuruhn...<br />
So reite denn, ich will dich nicht mehr fragen,<br />
Die weißen Wolken steigen und vergehn dort ohne Unterlaß.</p>
</div>
<p class="dichter">Wang Wei</p>
</body>
</html>

```

Das Design dieser Webseiten wird durch die Stylesheets in *gedichte.css* vorgegeben.

Listing 7.7: gedichte.css

```

body          { background-color: white; color: black;
                margin-top: 25px; margin-left: 25px;
                font-family: Arial, sans-serif;
                font-size: 12pt }
div.gedicht  { margin-left:35px }
p.dichter    { margin-left:300px }

```

Über den body-Stil geben wir die Farben und die Schrift vor. Wichtig ist auch die Angabe der oberen und unteren Ränder, damit der Text der Webseiten vom Rand des Frames abgerückt wird.

Der Text der Gedichte soll etwas weiter eingerückt werden. Aus diesem Grund wird jedes Gedicht in <div>-Tags eingeschlossen (erforderlich, weil es auch Gedichte mit mehreren Strophen gibt) und mittels des Stylesheets div.gedicht eingerückt.

Am Ende des Gedichts steht der Name des Dichters, der mit Hilfe des Stylesheets p.dichter weit nach rechts eingerückt wird.

Die Eingangsseite (index.html)

Zu guter Letzt setzen wir noch eine Eingangsseite zu unserem Web auf (siehe Abbildung 7.3).

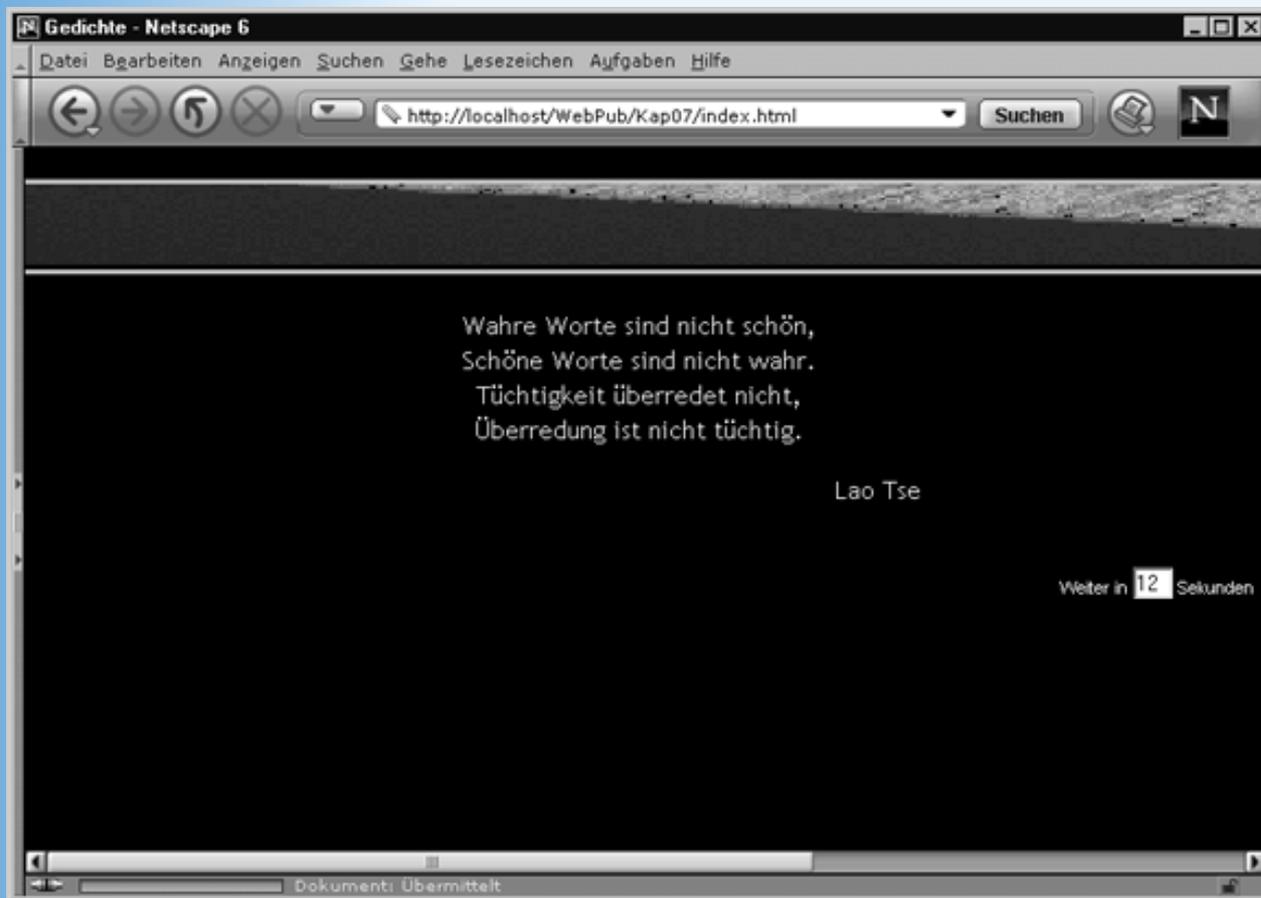


Abbildung 7.3: Die Eingangsseite

Wir speichern diese Datei unter *index.html*, damit sie automatisch aufgerufen wird, wenn der Websurfer als Adresse lediglich das Verzeichnis des Webs angibt.



*Gibt der Websurfer nur ein Zielverzeichnis an, sucht der Webserver in diesem Verzeichnis nach einer Datei *index.html*, *index.htm* oder *default.html*. Wie diese Standarddateien genau heißen müssen und in welcher Reihenfolge der Webserver nach ihnen sucht, hängt vom Webserver ab und kann gegebenenfalls beim zuständigen Webserver-Provider erfragt werden.*

Damit die Eingangsseite mit unserem Web konsistent ist, verwenden wir für die Eingangsseite das gleiche Design (Stylesheet-Datei *gedichte_eingang.css*) und das gleiche Banner wie für unser Web.

Zur Unterstützung von Suchrobotern fügen wir die gleichen META-Tags wie auf der Frameseite ein. Den zugehörigen Textabschnitt, in dem die Stichwörter aus dem <head>- Abschnitt auftauchen, fügen wir am Ende der Seite als eigenen Absatz ein. Mit Hilfe des Stylesheets

```
#verborgen { visibility: hidden; color: black }
```

sorgen wir dafür, dass dieser Abschnitt nicht sichtbar ist (er dient ja nur zur Zufriedenstellung der Suchroboter). Wir setzen dazu die Stileigenschaft *visibility* auf *hidden* und passen die Vordergrundfarbe der Hintergrundfarbe an (für Browser, die Stileigenschaft *visibility* nicht unterstützen).

Listing 7.8: index.html

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Gedichte</title>
  <!-- Suchmaschinen unterstützen -->
  <meta name="language" content="de">
  <meta lang="de" name="keywords" content="Gedichte,Rilke,Wang
    Wei,Longfellow, Dufu, Fontane">
  <meta name="description" content="Gedichtsammlung mit eigenen Werken
    und Gedichten ausgewählter Dichter (Schwerpunkt auf alten
    chinesischen Dichtern) ">
  <meta name="revisit-after" content="30 days">
  <meta name="robots" content="all">
  <link rel="start" type="text/html" href="frame_seite.html"
    title="Ausgewählte Gedichte">

  <!-- Stylesheets -->
  <meta http-equiv="Content-Style-Type" content="text/css">
  <link rel="StyleSheet" type="text/css" href="gedichte_ingang.css">
<!-- Ein wenig dynamisches HTML mit JavaScript statt

  <meta http-equiv="refresh" content="15; url="frame_seite.html">
-->
<script language="JavaScript">
function countdown()
{
  if (document.formular.meincounter.value > 0)
  {
    document.formular.meincounter.value =
      document.formular.meincounter.value-1;
    Timer = setTimeout("countdown()", 1000);
  }
  else
    location.href = "frame_seite.html";
}
</script>
</head>
<body onload="countdown()">
<div class="zentriert">
  
  <p>Wahre Worte sind nicht schön,<br />
    Schöne Worte sind nicht wahr.<br />
    Tüchtigkeit überredet nicht,<br />
    Überredung ist nicht tüchtig.</p>
  <p class="dichter">Lao Tse</p>
  <br />
  <div id="zaehler">
  <form name=formular>
    Weiter in <input type="text" name="meincounter" size="2"
      value="15"> Sekunden
  </form>
</div>

```

```
<p id="verborgen">Gedichtesammlung mit ausgewählten Gedichten von Rilke, Wang  
Wei, Longfellow, Dufu und Fontane</p>  
</body>  
</html>
```

Statt eines Hyperlinks zur Hauptseite unseres Webs haben wir uns den Luxus einer kleinen Spielerei erlaubt und eine automatische Weiterschaltung eingerichtet. Allerdings haben wir dazu nicht das übliche META-Tag verwendet, denn dann könnte der Websurfer ohne einen zusätzlich einzufügenden Hinweistext gar nicht erkennen, dass er automatisch weitergeleitet wird und verzweifelt nach einem Link suchen. Stattdessen haben wir ein JavaScript-Skript eingerichtet, dass in Zusammenarbeit mit einem Text-Steurelement einen Countdown erzeugt und am Ende des Countdowns für die Weiterleitung zur Hauptseite sorgt.

Sie brauchen das JavaScript-Skript nicht zu verstehen, betrachten Sie es einfach als Appetizer, der Ihnen Lust auf die nächste Woche machen soll.

7.2 Inspiration finden

Eine der Gründe für die Faszination des Internets ist seine anarchische Struktur und Zusammensetzung, die - bestenfalls - durch gewisse Höflichkeitsregeln eingeschränkt wird. Für Sie als Autor einer persönlichen Homepage bedeutet dies, dass Sie in der Wahl des Designs und des Inhalts Ihrer Webseiten völlig frei sind. Dies ist erfreulich und begrüßenswert, aber nicht unbedingt besonders hilfreich.

Stöbern Sie im Internet

Sofern Sie nicht schon eine klare Vorstellung vom Aufbau Ihrer persönlichen Homepage haben, nehmen Sie sich doch einfach ein wenig Zeit, um sich ins Internet einzuloggen und im WWW nach interessanten Homepages zu stöbern.



Adressen für private Homepages findet man in den Mitglieder-Verzeichnissen Ihrer Internet-Provider, über Suchmaschinen oder Freunde und Bekannte).



Abbildung 7.4: Mit Frames realisierte, liebevoll aufgebaute, mittelständische Website

Notieren Sie sich, welche Elemente häufig in Homepages verwendet werden und wie die verschiedenen Webs und deren Seiten aufgebaut sind. Versuchen Sie Modetrends auszumachen und lassen Sie sich von den Ideen anderer inspirieren. Merken Sie sich, was Ihnen im Einzelnen gefallen und was Ihnen missfallen hat.

Lassen Sie sich den Quelltext der einzelnen Webseiten anzeigen und lernen Sie daraus. (Aber Vorsicht! Viele Webseiten sind noch in altem HTML-Code geschrieben.)



Abbildung 7.5: Die Eingangsseite zur Website der Mailänder Scala

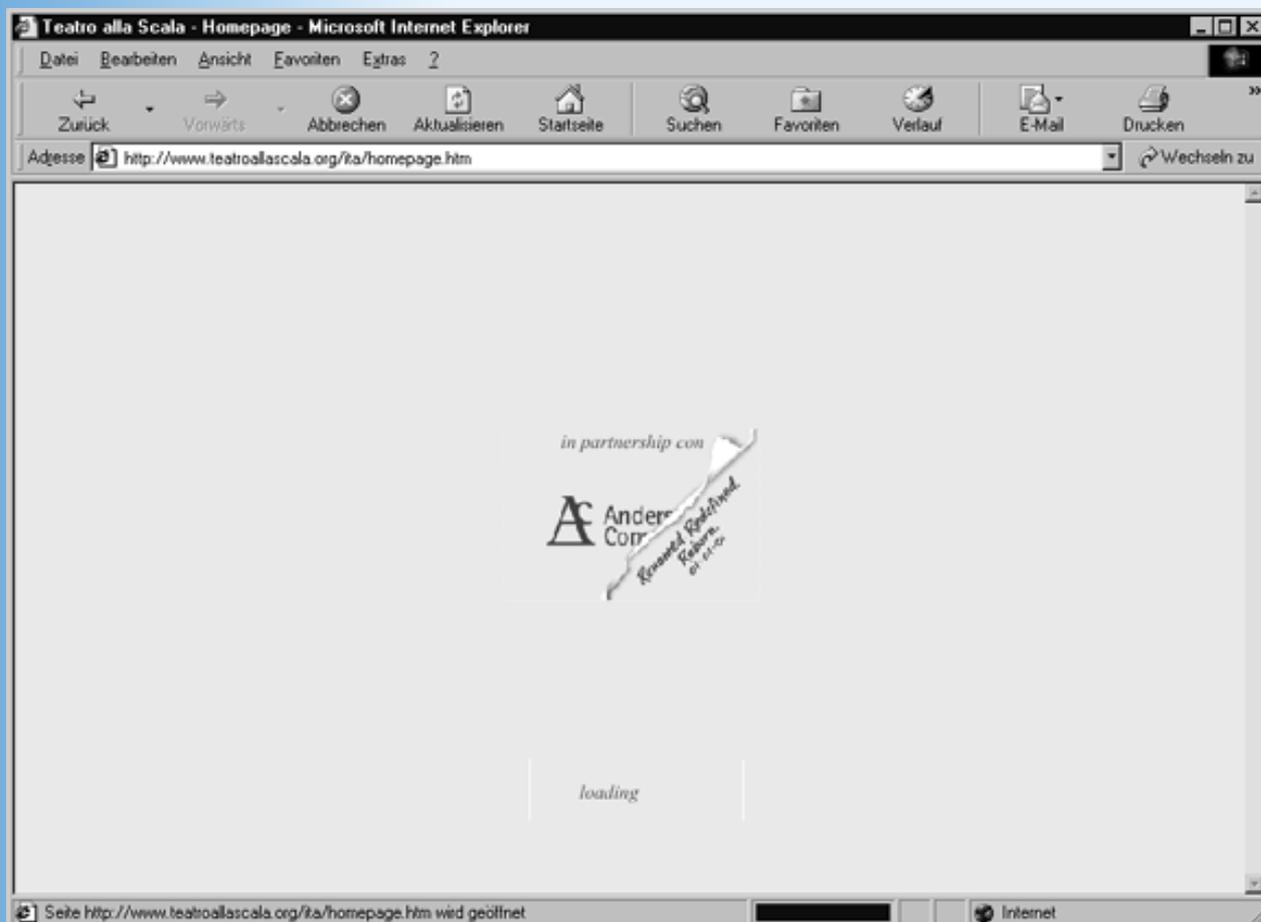


Abbildung 7.6: Tunnelseite

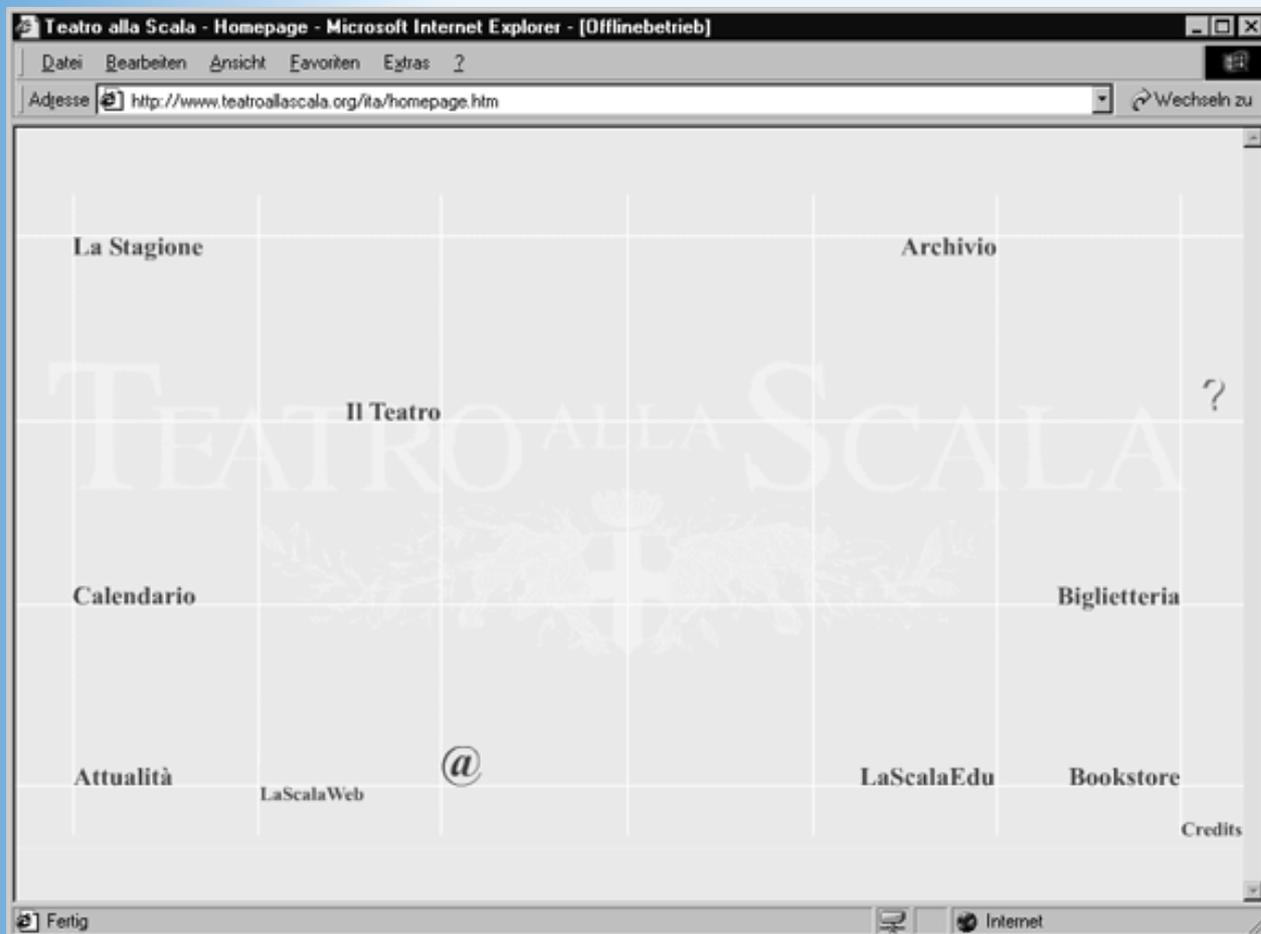


Abbildung 7.7: Eine Ansicht der Hauptseite

Urheberrecht im Internet

Wer sich nicht nur anregen lassen möchte, sondern Gefundenes am liebsten selbst verwenden möchten, der sollte beachten, dass es auch im Internet einen Urnehberschutz gibt.

- Werke sind im urheberrechtlichen Sinne geschützt
- Copyright-Zeichen werden nicht benötigt
- Ideen und Informationen sind nicht geschützt

In Deutschland sind alle Werke (Bilder, Fotos, Schriftstücke) einer Person urheberrechtlich geschützt. Das gilt auch für eine gestaltete Homepage (Copyrightzeichen sind daher nicht nötig). Ideen und Informationen sind dagegen nicht automatisch geschützt. Wenn Sie also auf Ihrer Webseite irgendeine attraktive Idee umsetzen (der Mauszeiger verwandelt sich in eine Maus), dann kann jemand diese Idee aufnehmen und mit einer Katze nachprogrammieren

- Datenbankinformationen und Linksammlungen sind geschützt

Reine Informationen, wie sie in der Presse gefunden werden können, sind nicht geschützt und könnten deshalb von jedem weiter verarbeitet werden, wenn er sie auf Ihrer Webseite findet. Datenbankinformationen sind dagegen wieder geschützt.

Auch Listen mit attraktiven Informationen, die Sie mühsam über Jahre gesammelt haben, sind geschützt. Da beginnt der Übergang zwischen Information und Datenbank. Eine lange Liste kann eine Datenbank darstellen und darf deshalb nicht einfach von einer fremdem Seite übernommen werden. Die beste Vorgehensweise, wenn Sie solche

Listen verwenden wollen, ist immer noch, den Autor zu fragen. Der fühlt sich dann durch eine nett geschriebene Email meist geehrt und wird seine Einwilligung gerne geben.

- Vorsicht bei Links und Frames

Links zu Seiten, die in Deutschland verbotene Inhalte darstellen, sind äußerst kritisch zu beurteilen und wurden schon gerichtlich untersagt.

Wer schon einmal mit Frames programmiert hat, weiß, wie leicht man eine fremde Seite über einen externen Link in einen Frame der eigenen Website einblenden kann. Der eigentliche Inhaber der Webseite wird dies aber wohl kaum gut heißen, wenn Sie unter Ihrem Layout seine Seite mit einbinden. Zumal wenn er mit viel Geld seine Seite von einem Grafiker professionell hat erstellen lassen und das ganze teure Layout in Ihrem Frame nicht zur Geltung kommt.

- Private Kopien sind erlaubt (für Gruppen < 20 Pers)
- Urheberrecht bis 70 Jahre nach Tod

Das deutsche Urheberrecht gilt für geistige Werke nur bis 70 Jahre nach dem Tod des Autors. Ältere Inhalte können Sie entsprechend ohne Nachfrage publizieren.

7.3 Todsünden des Webdesigns

Zum Abschluss habe ich noch eine Zusammenstellung von Fehlern, die Sie nicht machen sollten (wobei einige dieser Fehler tödlicher sind als andere):

1. Ladezeit sehr groß, der Websurfer sieht nichts (Falls nicht vermeidbar: Ladeseite einfügen, vergleiche Website der Mailänder Scala, Abbildung 7.6)
2. Werbebanner vor dem eigentlichen Inhalt
3. Eingang in den eigentlichen Inhalt nicht zu finden
4. Als erstes Formular ausfüllen, bevor es etwas zu sehen gibt.
5. Zuviel Text auf der Startseite
6. Durchgehende horizontale Linien zur Untergliederung einer Webseite
7. Häufiger Wechsel der Schrift. Noch schlimmer: Wechsel der Schriftart in einem Absatz (Alternative: Hervorhebung von Textpassagen durch Schriftstil (Farbe, Kursiv, Fett) und nicht durch Schriftart.
8. Verwendung von Leerzeichen und Leerzeilen für Einzüge (Alternative: margin, padding, text-indent)
9. Aufdringliche Hintergrundbilder, die in den Vordergrund drängen (Alternative: das Hintergrundbild aufhellen (bei schwarzer Schrift) oder anderen Hintergrund wählen.
10. Aliasing an Bildrändern
11. Unnötiger Einsatz von Javascript basierten Links (siehe Kapitel 8.2)
12. Unvermittelt neue Browserfenster öffnen lassen (Fensterdschungel)
13. Nerviger Hintergrundsound, der endlos dudelt.
14. Exzessiver Einsatz neuester Technologien, die auf den Systemen der meisten Websurfer nicht korrekt angezeigt werden können (Falls nicht vermeidbar: Links auf Website, wo man neuesten Browser oder Plug-In zur Unterstützung der betreffenden Technologien herunterladen kann)
15. Keine Alternative zur Framedarstellung
16. Zu viele Frames in einer Frame-Seite

7.4 Workshop

Heute gibt es keine Fragen und kein Quiz. Wer will kann die gewonnene Zeit nutzen, seiner Fantasie freien Lauf zu lassen und eine eigene Website zu kreieren.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

Tag 8

Wozu JavaScript?

Wer sich als Webautor überlegt, wie er mehr Interaktivität in seine Webseiten einbauen könnte, der wird nicht gleich an CGI oder Java denken. Die erste und meist die einfachste Option ist üblicherweise die Einbindung von JavaScript-Skripten. Was aber ist ein JavaScript-Skript? Was ist überhaupt JavaScript? Wie kann man JavaScript-Skripte in Webseiten einbinden, und was kann man eigentlich mit JavaScript so alles machen? Diesen Fragen wollen wir am heutigen Tag nachgehen.

Die Themen im Einzelnen:

- Was ist JavaScript?
- Wie wird JavaScript-Code verarbeitet?
- Was bedeutet es, dass JavaScript-Code clientseitig verarbeitet wird?
- Wo stammt JavaScript her und wer definiert den Umfang der Sprache?
- Wie bindet man JavaScript-Code in Webseiten ein?
- Wie wird Javascript-Code gestartet und ausgeführt?
- Was kann man mit JavaScript machen?
- Wie kann man JavaScript-Code auf Fehler untersuchen (debuggen)

8.1 Was ist JavaScript?

JavaScript ist eine Skriptsprache. Das soll heißen, dass JavaScript-Programme nicht kompiliert, sondern interpretiert werden.

Kompilation versus Interpretation

Der erste Schritt beim Programmieren besteht immer darin, den Quelltext des Programms aufzusetzen. Der Quelltext ist ein einfacher, unformatierter ASCII-Text, in dem man die Befehle niederschreibt, die der Rechner bei Ausführung des Programms abarbeiten soll - natürlich unter Verwendung der für die jeweils verwendete Programmiersprache üblichen Syntax.

Listing 8.1: Beispielquelltext für ein C++-Programm

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hallo Welt!" << endl;
    return 0;
}
```

Mit diesem Quelltext kann der Rechner allerdings nichts anfangen. Dazu ist ein zweiter Schritt

erforderlich, nämlich die Umwandlung des Quelltextes in Maschinencode (Maschinencode ist binärcodiert (Folge von Nullen und Einsen) und verwendet einen kleinen Satz elementarer Befehle, die der Prozessor des Rechners versteht).

Zur Umwandlung eines Quelltextes in Maschinencode gibt es zwei unterschiedliche Konzepte:

Kompilation

Hierzu wird ein spezielles Programm (der Compiler) aufgerufen, das den Quelltext des Programms in Maschinencode übersetzt. Das Ergebnis ist eine ausführbare Datei (EXE- Datei). Diese kann als eigenständiges Programm direkt aufgerufen und ausgeführt werden.

- Kompilierte Programme sind meist schneller und effizienter als vergleichbare interpretierte Programme.
- Kompilierte Programme sind nicht portabel, da der vom Compiler erzeugte Maschinencode immer auf einen bestimmten Prozessortyp abgestimmt ist und von anderen Prozessoren nicht verarbeitet werden kann.
- Beispiele für kompilierte Programmiersprachen sind Pascal, C/C++ und in gewisser Weise auch Java.

Interpretation

Hier fallen Übersetzung und Ausführung des Programms zusammen. Dies leistet der Interpreter. Er liest das Programm Zeile für Zeile ein, erzeugt entsprechenden Maschinencode und lässt diesen direkt ausführen.

- Interpretierte Programme sind in der Ausführung langsamer als vergleichbare kompilierte Programme, da im Hintergrund ja immer der Interpreter läuft und die Zeit für die Umsetzung in Maschinencode hinzugerechnet werden muss.
- Interpretierte Programme können problemlos portiert werden, da sie nur als Quelltext vorliegen. Voraussetzung ist allerdings, dass auf den Rechnern, auf denen die Programme interpretiert und ausgeführt werden sollen, ein passender Interpreter installiert ist.
- Beispiele für interpretierte Programmiersprachen sind Basic, Perl, Java und eben auch JavaScript.

Clientseitig versus serverseitig

Bei der Programmierung für Webseiten muss man zwischen serverseitiger und clientseitiger Programmierung unterscheiden. Vor JavaScript konnte man nur die serverseitige Programmierung, bei der die Programme auf dem Webserver-Rechner ausgeführt werden, auf dem auch die Webseiten liegen. Dies sah und sieht dann beispielsweise so aus:

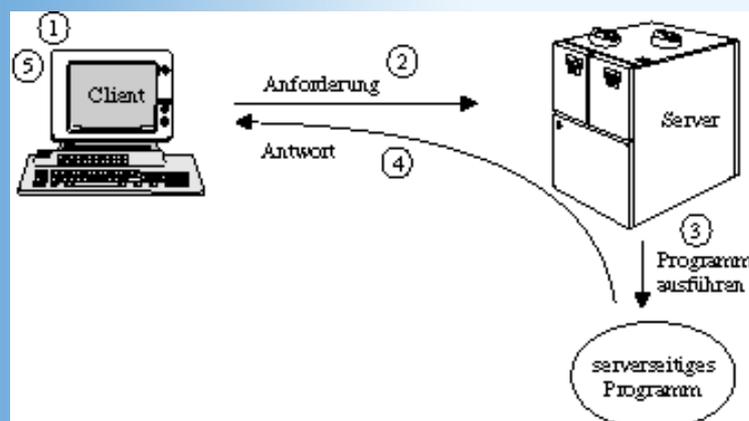


Abbildung 8.1: Ausführung eines serverseitigen Programms

1. Der Webbesucher füllt ein Formular aus und klickt auf den Abschicken-Schalter des Formulars. Der Autor der Webseite hat den Abschicken-Schalter des Formulars mit dem URL eines serverseitigen Programms verbunden.
2. Der Webbrowser schickt dem Webserver, der in dem URL angegeben ist, eine Aufforderung, dass Programm aufzurufen und diesem die Formulardaten zu übergeben.
3. Der Webserver ruft das Programm auf. Das Programm wird auf dem Server-Rechner ausgeführt und verarbeitet die Formulardaten. Als Ergebnis erzeugt es beispielsweise den HTML-Code einer Antwortseite.
4. Der Webserver schickt die vom Programm generierte Webseite an den Browser zurück.
5. Der Webbesucher sieht in seinem Browser die Bestätigung, dass seine Formulardaten angekommen sind und verarbeitet wurden.

Diese Form der Webseitenunterstützung durch Programme ist äußerst mächtig und aus der heutigen Webtechnologie nicht mehr wegzudenken, und dennoch gibt es auch Nachteile, die mit ihr verbunden sind.

- Die Ausführung ist aufgrund der Datenübertragung über das Netz relativ langsam.
- Greifen Hunderte oder gar Tausende von Websurfern gleichzeitig auf das Programm auf dem Server zu, wird es hundert- oder tausendmal parallel auf dem Server-Rechner ausgeführt und kann diesen unter Umständen lahm legen.
- Programme, die auf dem Webserver ausgeführt werden, stellen immer ein Sicherheitsrisiko für den Server-Rechner dar, zumal die Programme Eingaben verarbeiten, die von beliebigen (unter Umständen böswilligen) Internet-Teilnehmern kommen können. Aus diesem Grunde unterliegt die Installation serverseitiger Webprogramme gewissen Sicherheitsbeschränkungen, und Webautoren müssen sich diesbezüglich meist an den zuständigen Server-Administrator wenden.
- Der Datenaustausch zwischen Webbrowser, Webserver und serverseitigem Programm unterliegt bestimmten strengen Regeln, die in der CGI-Spezifikation festgelegt sind (siehe Kapitel 17).

Kurz gesagt: Die serverseitige Programmierung ist schwerfällig und sehr technisiert. Viele Webdesigner suchen hingegen das genaue Gegenteil: einen Weg, wie man Webseiten ohne jahrelange Programmiererfahrung und großen technischen Aufwand dynamischer und lebendiger machen kann. Sie finden ihn in der clientseitigen Programmierung mit JavaScript.

Bei der Programmierung mit JavaScript schreibt der Webdesigner den Programmcode direkt in den HTML-Code der Webseite (oder in eine externe Datei, die über einen src- Verweis eingebunden wird). Der JavaScript-Code wird mit der Webseite auf die Rechner der Webbesucher (zur Unterscheidung vom Server-Rechner spricht man in diesem Falle von den Client-Rechnern) heruntergeladen und dort mit Hilfe eines Interpreters ausgeführt. Da der Programmcode direkt auf den Client-Rechnern ausgeführt wird, ist die Ausführung viel schneller, der Webserver wird nicht weiter belastet und, weil die Programme auch kein Sicherheitsrisiko mehr für den Webserver darstellen, entfällt der ganze Verwaltungsaufwand, der mit der serverseitigen Programmierung verbunden ist.

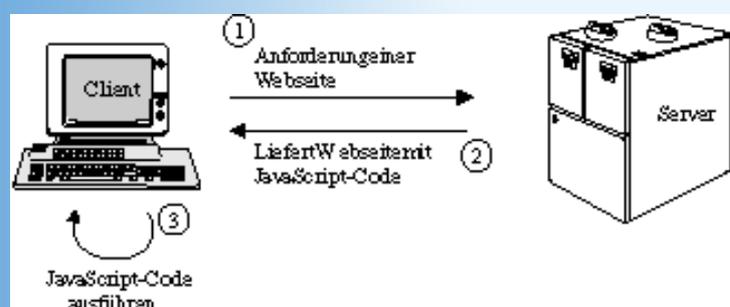


Abbildung 8.2: Ausführung eines clientseitigen Programms

Offensichtlich ist JavaScript der serverseitigen Programmierung gerade in den Punkten überlegen, die dieser zum Nachteil gereichen. Da könnte sich die Frage aufdrängen, ob man nicht ganz zugunsten der clientseitigen Programmierung mit JavaScript auf die serverseitige Programmierung verzichten sollte? Die klare Antwort ist: nein, clientseitige Programmierung mit JavaScript ist kein Ersatz für die serverseitige Programmierung, sondern eine Ergänzung. Die clientseitige und die serverseitige Webprogrammierung beruhen auf grundverschiedenen technischen Voraussetzungen, die jeweils eigene Vor- und Nachteile und Einsatzgebiete mit sich bringen. Für dynamische Effekte und schnelle Interaktion mit dem Webbesucher eignet sich vor allem die clientseitige Programmierung mit JavaScript, für anspruchsvollere Aufgaben oder das Zurückliefern von Daten an den Webserver (beispielsweise Bestellungen und Aufträge in E-Commerce-Systemen) ist die serverseitige Programmierung unersetzlich.

JavaScript und der Browser

Da JavaScript eine interpretierte Sprache ist, bedarf es zur Ausführung von JavaScript-Code eines Interpreters. Anders ausgedrückt: Wenn jemand, der auf seinem System keinen JavaScript-Interpreter installiert hat, in seinem Browser Ihre mit JavaScript aufgepeppte Webseite anschaut, kann er von den ganzen dynamischen JavaScript-Effekten rein gar nichts sehen.

Sollten wir also beim Einsatz von JavaScript Vorsicht walten lassen und eventuell gar auf das Skripting verzichten? Nein, denn schon seit mehreren Generationen werden die gängigsten Browser mit integriertem JavaScript-Interpreter ausgeliefert. Es dürfte also kaum noch Internet-Nutzer geben, die mit Browsern ohne JavaScript-Unterstützung surfen.

Über mangelnde Unterstützung seitens der Browser können wir uns also nicht beklagen, wohl aber über mangelnde Einheitlichkeit, denn JavaScript im Internet Explorer ist nicht dasselbe wie JavaScript im Netscape Navigator (oder irgendeinem anderen Browser). Dies hat unter anderem historische Gründe.

Geschichte

JavaScript wurde von Netscape Communications Corp. entwickelt und 1995 zum ersten Mal vorgestellt (damals noch unter dem Namen LiveScript). Durch die Integration des JavaScript-Interpreters in den damals marktführenden Netscape Navigator 2 sorgte Netscape für eine schnelle und weite Verbreitung - Grundvoraussetzung dafür, dass die Sprache auch von den Webdesignern akzeptiert wurde. Diese musste man denn auch nicht lange bitten, sie stürzten sich begierig auf die (damals noch recht beschränkte) neue Technologie, mit deren Hilfe man zum ersten Mal in der Geschichte des Webs clientseitigen Code zur Unterstützung von Webseiten aufsetzen konnte.

Angesichts der wachsenden Popularität und des Potentials, das in dieser Technologie steckte, dauerte es nicht lange, bis Microsoft zusammen mit den Internet Explorer 3 seinen eigenen JavaScript-Dialekt präsentierte (der aus lizenzrechtlichen Gründen JScript heißt). Beide Browser-Hersteller entwickelten ihre JavaScript-Dialekte beständig weiter, immer hin- und hergerissen zwischen der Unterstützung des jeweils anderen Dialekts und der Einführung eigener, proprietärer Erweiterungen.

Um zu verhindern, dass die beiden Dialekte zu weit auseinander driften, begann das europäische ECMA-Institut bereits Ende 1996 damit, auf der Grundlage von Netscape- JavaScript und Microsoft-JScript einen allgemeinen Skript-Standard zu erarbeiten. Dieser wurde 1997 zum offiziellen ECMA-Standard, 1998 wurde er zudem als ISO/IEC-Standard anerkannt. Seitdem wurden sowohl die Dialekte

als auch der Standard stetig verbessert und erweitert, der Kern der Sprache ist mittlerweile aber recht stabil und einheitlich.

Der ECMA-Standard regelt nur die Syntax der Sprache, mindestens ebenso wichtig für den Webdesigner und JavaScript-Programmierer¹ ist allerdings der Zugriff von JavaScript- Programmen aus auf die HTML-Elemente der Webseiten. Für diese Schnittstelle wurde erst im Oktober 1998 eine offizielle Empfehlung ausgesprochen (DOM 1-Referenz des W3C), und so ist es nicht verwunderlich, dass es hier gravierende Unterschiede zwischen den Browsern gibt. Wir werden uns an gegebener Stelle noch ausführlicher damit beschäftigen müssen.

8.2 JavaScript-Code in Webseiten einbinden

JavaScript-Code kann über das Tag `<script>` oder eines der in HTML definierten Ereignisattribute (onload, etc.) eingebunden werden.

Das `<script>`-Tag

Wie bereits erwähnt, kann man JavaScript-Code direkt in den HTML-Code der Webseite schreiben, man muss den Code lediglich in `<script>`-Tags einbinden, um dem Browser mitzuteilen, dass nun kein anzuzeigender Text, sondern ein Skript folgt.

```
<script type="text/javascript">
  . . . hier steht der Skript-Code ...
</script>
```

Das Attribut `type="text/javascript"` ist übrigens für jedes `<script>`-Tag zwingend vorgeschrieben, da es neben JavaScript noch andere Skriptsprachen für die clientseitige Programmierung gibt. Über das `type`-Attribut teilen wir dem Browser mit, welche Skriptsprache wir verwenden und welchen Interpreter er zur Ausführung des Skript-Codes aufrufen muss. Mögliche Werte für das `type`-Attribut sind: `text/javascript`, `text/vbscript` und `text/tcl`.



Früher wurde zu diesem Zweck meist das `language`-Attribut verwendet, doch ist dessen Gebrauch mittlerweile »deprecated«.

JavaScript-Code kann man übrigens überall im HTML-Code einfügen - im Header- wie im Body-Abschnitt. Für welchen Ort man sich entscheidet, hängt meist davon ab, was man mit dem JavaScript-Code machen will (siehe nachfolgender Abschnitt).

Ausführung und Ort der Deklaration

Es gibt drei Wege wie JavaScript-Code ausgeführt werden kann:

- automatisch beim Aufbau der Webseite im Browser
- bei Eintritt eines bestimmten Ereignisses
- bei Anklicken eines Links

Automatische Ausführung beim Laden der Webseite

Lädt der Browser eine Webseite mit Skriptcode geht er sämtliche `<script>`-Tags durch und schaut, ob diese ausführbaren Code enthalten. Wenn ja, führt er diesen aus. Man kann dies auf unterschiedliche Weise nutzen.

Eine Möglichkeit ist zum Beispiel per JavaScript HTML-Code in die Webseite einzufügen. In diesem Fall fügt man den Skriptcode im Body-Abschnitt an der Stelle ein, wo später die HTML-Ausgabe des Skripts stehen soll.

Listing 8.2: script1.html - automatische Ausführung von Skriptcode

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>JavaScript</title>
</head>
<body>
<h1>Und jetzt der JavaScript-Code:</h1>
<script type="text/javascript">
  document.write("<p>Hier spricht Ihr <em>JavaScript!<em><p>");
</script>
</body>
</html>
```



Abbildung 8.3: JavaScript-generierte Ausgabe

Man kann die automatische Ausführung aber auch dazu nutzen, beim Laden der Webseite irgendwelche Arbeiten zu erledigen, die keine direkt sichtbare Ausgabe erzeugen, beispielsweise Cookies einzurichten oder vorab schon einmal die Bilder für eine später auszuführende JavaScript-Animation zu laden. In diesem Falle empfiehlt es sich, den Skript-Code der besseren Übersicht wegen in den Header-Abschnitt einzufügen.



In JavaScript wird streng zwischen Groß- und Kleinschreibung unterschieden. Wenn wir in obigem Beispiel Document statt document schreiben würden, wäre der Code fehlerhaft!

Ausführung bei Eintritt eines bestimmten Ereignisses

Neben der automatischen Ausführung gibt es die Möglichkeit, JavaScript-Code bei Eintritt eines bestimmten Ereignisses ausführen zu lassen. Die HTML-Spezifikation definiert zu diesem Zweck eine ganze Reihe von Ereignissen, beispielsweise:

- die Webseite wird in den Browser geladen (onload)
- ein HTML-Element wurde angeklickt (onclick)
- der Mauszeiger wurde über ein Bild bewegt (onmousemove)
- der Wert eines Eingabefeldes hat sich geändert (onchange)
- die Webseite wird verlassen (onunload)

Aufgabe des Browser ist es, alle diese Ereignisse zu überwachen. Tritt eines dieser Ereignisse ein, prüft der Browser, ob das Ereignis vom Autor der Webseite mit JavaScript- Code verbunden wurde. Wenn ja, führt er diesen aus.

Ein Ereignis mit JavaScript-Code zu verbinden, ist nicht schwer. Man muss nur wissen, wie das zugehörige HTML-Attribut heißt und auf welche HTML-Tags man es anwenden kann. Das Attribut zu dem Laden-Ereignis der Webseite heißt beispielsweise onload und ist nur für das <body>-Tag definiert.



Eine Übersicht über die von HTML definierten Ereignis-Attribute, die übrigens alle mit »on« anfangen, finden Sie in Kapitel 10.1.

Möchte man beispielsweise per JavaScript den Hintergrund einer Webseite beim Laden blau einfärben, schreibt man:

```
<body onload="document.bgColor = 'blue' ;">
```

Möchte man umfangreicheren Code mit einem Ereignis verbinden, ist die obige Syntax allerdings recht unbequem und unübersichtlich. Als Ausweg lagert man den Skriptcode in <script>-Tags aus und schließt ihn in eine Funktionsdeklaration ein:

```
<script type="text/javascript">
function fkt_name()
{
  document.bgColor = "cyan";    // zyanblauer Seitenhintergrund
}
</script>
```

Die Funktionsdeklaration sorgt dafür, dass der Code - obwohl er jetzt in <script>-Tags steht - nicht automatisch beim Laden der Webseite ausgeführt wird. Gleichzeitig wird der Code mit einem Funktionsnamen verbunden und kann über diesen aufgerufen beziehungsweise mit einem Ereignis verbunden werden.

```
<body onload="fkt_name()">
```

Wenn Sie JavaScript-Code mit einem Ereignis verbinden, sollten Sie zwei Dinge beherzigen:

- Sie sollten dem Browser vorab mit Hilfe eines META-Tags angeben, welche Skriptsprache Sie verwenden.

```
<head>
...
<meta http-equiv="Content-Script-Type"
      content="text/javascript" />
</head>
```

Im Übrigen ist es keine schlechte Angewohnheit, für jede Webseite, die Skriptcode enthält, die Standardskriptsprache anzugeben (besser als die Angabe laufend zu vergessen und es der Intelligenz des Browsers zu überlassen, den richtigen Interpreter zu finden).

- Umfasst der Ereignisbehandlungscode mehr als ein (oder zwei) Anweisungen sollten Sie den Code in <script>-Tags auslagern und der besseren Übersichtlichkeit halber in den Header der Webseite einfügen.



Grundsätzlich ist es zu empfehlen, den Skriptcode einer Webseite so weit es geht, zentral im <head>-Abschnitt zu verwalten. Dies vereinfacht die Bearbeitung und Wartung der Webseite. Verbindlich ist diese Regel allerdings nicht, und im Falle von Skripten, die eine HTML-Ausgabe erzeugen (siehe oben), meist auch gar nicht möglich.

Listing 8.3: script2.html - Skriptcode zur Ereignisbehandlung

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>

<title>JavaScript</title>
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
  function farbe_setzen()
  {
    document.bgColor = "cyan";
  }
</script>
```

```

</script>
</head>
<body onload="farbe_setzen()">
<h1>JavaScript</h1>
<p>Wenn Sie eine schwarze Schrift vor zyanblauem Hintergrund sehen, wurde der
JavaScript-Code korrekt ausgeführt.</p>
</body>
</html>

```

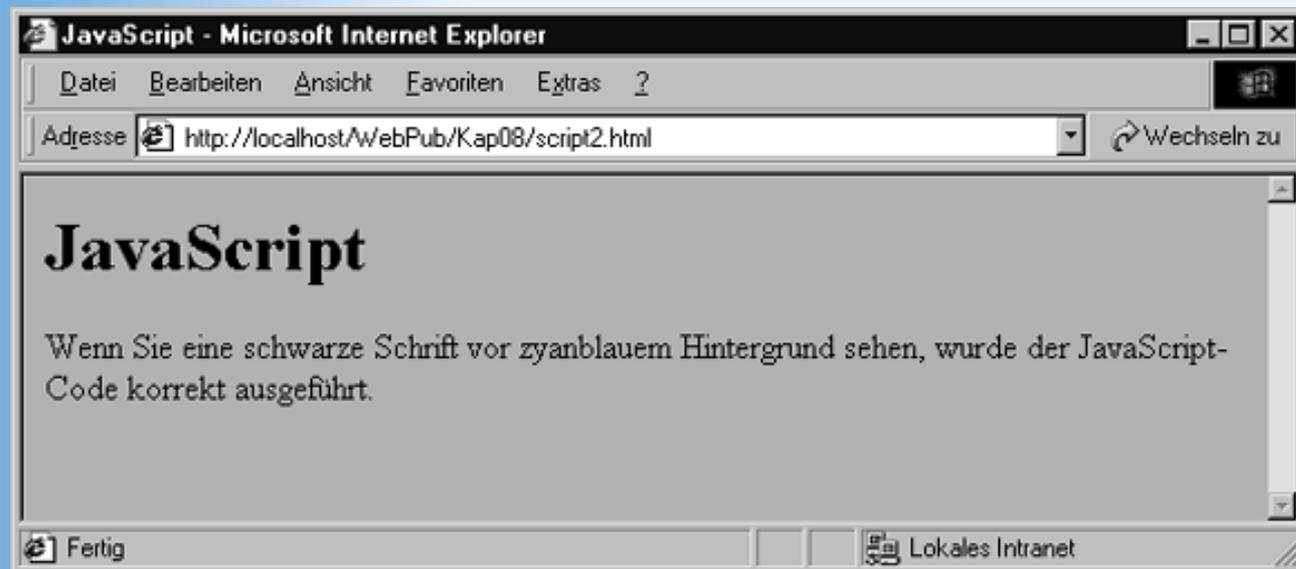


Abbildung 8.4: Mit JavaScript gesetzte Farben

JavaScript-Code als Ziel eines Hyperlinks

Schließlich besteht die Möglichkeit, einen Hyperlink statt mit einer neuen Webseite mit einer JavaScript-Funktion zu verbinden. Sie müssen dazu als Wert für das href-Attribut lediglich das Schlüsselwort javascript: und den Funktionsnamen angeben.



Nicht den Doppelpunkt hinter javascript vergessen!

Listing 8.4: script3.html - Hyperlink mit JavaScript-Funktion verbinden

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>

<title>JavaScript</title>
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
function farbe_setzen()
{
document.bgColor = "cyan";

```

```

    }
</script>
</head>
<body>
<h1>JavaScript</h1>
<p>Wenn Sie <a href="javascript: farbe_setzen()">hier klicken</a> wird der
Hintergrund zyanblau.</p>
</body>
</html>

```



Es ist allerdings verpönt, Hyperlinks mit JavaScript-Funktionen zu verbinden, die - wie in obigem Beispiel - nur kleine Veränderungen oder Manipulationen vornehmen. Dies liegt daran, dass Websurfer beim Klick auf einen Hyperlink üblicherweise erwarten, dass eine neue Seite geladen wird. Diese Erwartung sollte man respektieren und nicht mutwillig enttäuschen (Ausnahmen bestätigen die Regel). Nutzen Sie diese Form des JavaScript-Codeaufrufs also vornehmlich, um den Aufruf einer Webseite mit der Ausführung von JavaScript-Code zu verbinden (beispielsweise um je nach Browser unterschiedliche Webseiten zu laden, siehe Kapitel 10.5, oder um intern zu zählen, wie oft der Hyperlink angeklickt wird, etc. Am Ende des Codes der JavaScript-Funktion laden Sie dann die neue Webseite: `location.href = "neueWebseite.html"` (siehe Kapitel 9.6.2).

JavaScript in Browsern ohne JavaScript-Unterstützung

Mittlerweile dürfte der Anteil an Websurfern, die mit Browsern ohne JavaScript-Unterstützung durchs Internet surfen, ziemlich gering sein, doch früher war dies anders. Aus diesem Grunde wurden verschiedene Techniken entwickelt, wie man sicherstellen kann, dass Webseiten mit JavaScript in solchen Browsern wenigstens halbwegs adäquat dargestellt werden.

Skriptcode verbergen

Das erste Problem ist, dass ein Browser ohne Skriptunterstützung auch das `<script>`-Tag nicht erkennen wird. Wie aber verfährt ein anständiger Browser mit Tags, die er nicht kennt? Er versucht einfach, den Inhalt des Tags irgendwie darzustellen. Im Falle der `<script>`-Tags bedeutet dies, dass er den Skript-Code als einfachen Text in der Webseite ausgeben wird.

Dies ist natürlich überhaupt nicht erwünscht und als Ausweg ist man dazu übergegangen, Skript-Code auszukommentieren.

```

<script type="text/javascript">
<!--
function fkt_name()
{
    document.bgColor = "cyan";
}
// -->
</script>

```

Die Zeichenfolge <!-- leitet einen HTML-Kommentar ein. Browser ohne Skriptunterstützung werden den nachfolgenden Inhalt also ignorieren. Browser mit Skriptunterstützung stoßen dagegen gar nicht erst auf das Kommentarzeichen, weil sie zur Bearbeitung des Inhalts der <script>-Tags ja den Skript-Interpreter aufrufen (und dieser wiederum ignoriert das HTML-Kommentarzeichen).

Am Ende des Skripts wird der HTML-Kommentar mit --> beendet. Wichtig ist, dass diese Zeichenfolge mit Hilfe von // vor dem Skript-Interpreter verborgen wird (// leitet einen einzeiligen JavaScript-Kommentar ein, /* und */ schließen in JavaScript mehrzeilige Kommentare ein).



Wer seine Webseiten in Hinblick auf die fortschreitende Bedeutung von XML nach dem XHTML-Standard aufsetzen möchte (siehe Kapitel 21), stößt hier auf ein Problem. HTML-Kommentare können von XML-Parsern nicht nur ignoriert, sondern unter Umständen ganz entfernt werden - der Skriptcode würde auf diese Weise verloren gehen. Zudem interpretiert der XML-Parser die Zeichen < und & als HTML-Zeichen und nicht als Skript-Code. Letzteres kann man verhindern, indem man den Skriptcode in die Zeichenfolgen <![CDATA[und]]> einschließt, doch geht dadurch die Kompatibilität zu HTML verloren. Die beste Alternative für Autoren, die beiden Standards gerecht werden wollen, ist daher die Verwendung externer Skripts (siehe unten).

Alternativen HTML-Text anzeigen

Ein anderes Problem entsteht, wenn Sie im <body>-Abschnitt Skripte verwenden, die dynamische HTML-Elemente erzeugen und ausgeben (siehe Listing 8.2). In Browsern ohne Skriptunterstützung würden diese Ausgaben fehlen, und der Besucher Ihrer Website wird sich über den konfusen Aufbau und Text der Webseite wundern.

Für solche Fälle sieht der HTML-Standard das <noscript>-Tag vor, mit dem Sie einen alternativen Text, einen Link zu einer alternativen Webseite oder einen Hinweis ausgeben können, dass zur korrekten Darstellung dieser Webseite ein Browser mit JavaScript-Unterstützung benötigt wird.

```
<body>
<h1>Und jetzt der JavaScript-Code:</h1>
<script type="text/javascript">
  document.write("<p>Hier spricht Ihr <em>JavaScript!<em><p>");
</script>
<noscript>
  <p>Für diese Webseite ist ein Browser mit JavaScript-Unterstützung
    erforderlich! </p>
</noscript>
</body>
```



Manche Browser/Browser-Versionen kann man so konfigurieren, dass sie trotz vorhandenen JavaScript-Interpreters keinen JavaScript-Code ausführen sollen. Solchermaßen konfigurierte Browser werden ebenfalls den Inhalt der <noscript>-Tags anzeigen.

Externe Skripte

Skripte können auch als eigene Dateien mit der Extension .js auf dem Server abgelegt werden. Den URL des Skripts geben Sie dann im <script>-Tag an:

```
<script type="text/javascript" src="farben.js"></script>
```

Unter Verwendung einer JS-Datei würde der Code der Webseite aus Listing 8.3 beispielsweise wie folgt aussehen:

Listing 8.5: script4.html - Webseite, die JavaScript-Code aus einer Datei lädt

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>

<title>JavaScript</title>
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript" src="farben.js"></script>
</head>
<body onload="farben_setzen()">
<h1>JavaScript</h1>
<p>Wenn Sie eine schwarze Schrift vor zyanblauem Hintergrund sehen, wurde der
JavaScript-Code korrekt ausgeführt.</p>
</body>
</html>
```

Die dazugehörige JavaScript-Datei sehen Sie in Listing 8.6. Sie wird wie die Webseiten als reine Textdatei aufgesetzt und mit der Extension .js abgespeichert (wenn Sie Probleme mit dem Abspeichern haben, schauen Sie noch einmal in Kapitel 1.2 rein).

Listing 8.6: farben.js - einfache Textdatei mit JavaScript-Code

```
function farben_setzen()
{
  document.bgColor = "cyan";
}
```

Die Verwendung externer Skripte ist zwar mit etwas mehr Verwaltungsarbeit verbunden, bringt aber auch einige Vorteile mit sich:

- HTML-Code und Skript-Code sind sauber getrennt.
- Skript-Dateien können von mehreren Webseiten verwendet werden
- Skript-Dateien können einfach kopiert und für andere Webseiten überarbeitet werden
- Bei Verwendung externer Skripte gibt es keine Probleme mit dem Verbergen des Skriptcodes vor älteren Browsern ohne JavaScript-Unterstützung.

- Bei Verwendung externer Skripte gibt es keine Probleme bei der Umstellung von HTML zu XHTML.



Manche Webserver müssen erst für .js konfiguriert werden (MIME-Typ setzen).

8.3 Was kann man mit JavaScript anfangen - ein Schnupperkurs

Der folgende Programmierkurs ist als Schnupperkurs für den Einstieg in die JavaScript-Programmierung gedacht und soll Sie anhand einiger kleinerer Beispiele mit den wichtigsten JavaScript-Techniken und -Möglichkeiten vertraut machen. Die Beispiele sind möglichst einfach gehalten, damit man sie auch als Einsteiger ohne grundlegende Kenntnisse der JavaScript-Syntax nachvollziehen kann. Später, wenn wir uns mit der JavaScript-Syntax vertraut gemacht haben, werden wir viele der hier vorgestellten Techniken wieder aufgreifen und in praxisnahen Beispielen einsetzen.

Dynamischen Text ausgeben

Wie man mit Hilfe der JavaScript-Methode `document.write` einen HTML-Code in eine Webseite schreiben kann, haben Sie bereits gesehen:

Listing 8.7: text.html - dynamischen Text ausgeben

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Dynamischer Text</title>
</head>
<body>
<script type="text/javascript">
  document.write("<h2>Hallo</h2>");
  document.write("<p style='padding-left: 2em'>Hier spricht Ihr
    ↪<em>JavaScript!</em><p>");
</script>
</body>
</html>
```

Schauen wir uns diesen Skriptcode ein wenig genauer an. Zur Ausgabe wird hier die Konstruktion `document.write` verwendet. `document` ist ein JavaScript-Objekt, das die aktuelle Webseite repräsentiert. Es wird vom Browser erzeugt und verfügt über verschiedene Eigenschaften und Methoden², über die man auf die Webseite Einfluss nehmen kann. Eine dieser Methoden ist `write()`, mit deren Hilfe man HTML-Code in die Webseite einfügen kann.

Den auszugebenden HTML-Code kann man direkt in den runden Klammern der `write`-Methode übergeben, muss ihn aber dann in Anführungszeichen setzen. Komplikationen gibt es, wenn der auszugebende Text ebenfalls Anführungszeichen enthalten muss, beispielsweise wenn man wie in

obigem Beispiel ein Attribut (dessen Wert ja gemäß HTML 4 in Anführungszeichen stehen soll) mit ausgeben will. In so einem Fall kann man sich dadurch behelfen, dass man im auszugebenden HTML-Code einfache statt doppelte Anführungszeichen verwendet, den Anführungszeichen das Escape-Zeichen \ voranstellt oder den auszugebenden HTML-Code zuerst in einem String-Objekt ablegt (siehe Kapitel 9.6.1) und dann der write-Methode das String-Objekt übergibt.



Der Navigator 4 verarbeitet übrigens obigen Code nicht ganz korrekt. Er bezieht die style-Information aus unerfindlichen Gründen auf die Inline-Passage statt auf den ganzen <p>-Absatz. Im Netscape 6-Browser tritt dieser Fehler nicht auf.

JavaScript lediglich zur Ausgabe von HTML-Code in eine Webseite zu benutzen, ist ziemlich unnütz. Gepaart mit anderen JavaScript-Techniken ergeben sich aber interessante Möglichkeiten:

- Sie können mit JavaScript dynamische Inhalte ausgeben (beispielsweise berechnete Werte oder das aktuelle Datum, siehe nächster Abschnitt).
- Sie können mit JavaScript dynamisch ganz neue Webseiten erzeugen und anzeigen lassen (siehe Abschnitt 8.3.5).
- Sie können mit JavaScript-Cookies (siehe Kapitel 13) Daten zwischen Webseiten austauschen und den Text/Inhalt einzelner Webseiten an diese Daten anpassen.
- Schließlich besteht auch die Möglichkeit, mit JavaScript den Text eines HTML- Elements (eines <p>-Absatzes oder einer Textpassage in , oder ähnlichen Inline-Tags zu verändern. Dies geschieht dann aber nicht mehr über document.write, sondern über die DOM-Eigenschaft nodeValue (im Internet Explorer auch über die Eigenschaft innerHTML). Mehr dazu in Kapitel 10.4.6.

Aktuelles Datum ausgeben

Die Ausgabe von Text mit JavaScript wird erst so richtig interessant, wenn der Text dynamische Inhalte widerspiegelt - beispielsweise das aktuelle Datum. JavaScript definiert hierzu die Klasse Date.

Listing 8.8: datum.html - Ausgabe des aktuellen Datums

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Datumausgabe</title>
</head>
<body>
<p>Es ist
<script type="text/javascript">
  var Datumsobjekt = new Date();
  var datum = Datumsobjekt.toLocaleString();
  document.write(datum);
</script>
, und draußen ist ein schöner Tag.
```

```
</p>  
</body>  
</html>
```

Im vorangehenden Abschnitt haben Sie bereits das document-Objekt kennen gelernt, dass die aktuelle Webseite präsentiert. JavaScript-Objekte kommen nicht aus dem Nichts, vielmehr gehen Sie auf eine formale Klassendefinition zurück. In der Klassendefinition ist festgelegt, welche Eigenschaften und Methoden zu den Objekten der Klasse gehören. Die Klasse ist also so etwas wie der Bauplan für die Erzeugung ihrer Objekte.

Man kann dies mit der Errichtung von Fertighäusern vergleichen. Ein Architekt entwirft den Bauplan für das Fertighaus. Nach diesem Bauplan können dann jederzeit wirkliche Fertighäuser errichtet werden. Alle nach dem Bauplan errichteten Fertighäuser sehen grundsätzlich gleich aus, können sich aber durch individuelle Merkmale (Anstrich, Carport, Heizungssystem) unterscheiden. Übertragen auf JavaScript entspricht der Bauplan der Definition einer Klasse und die einzelnen Fertighäuser den Objekten der Klasse. Individualität erhalten die Objekte einer Klasse dadurch, dass ihre Eigenschaften unterschiedliche Werte annehmen können³.

Im Falle des document-Objekts sehen wir nichts von der zugehörigen Klasse, weil uns das document-Objekt fertig vom Browser zur Verfügung gestellt wird. Wenn wir auf das aktuelle Datum zugreifen wollen, müssen wir uns dagegen zuerst ein eigenes Objekt der Klasse Date erzeugen.

```
var Datumsobjekt = new Date();
```

Hier wird mit Hilfe des Schlüsselworts new ein neues Date-Objekt erzeugt. Das neue Objekt wird mit der Variablen Datumsobjekt verbunden, über die wir im Skript-Code weiter auf das Objekt zugreifen können.

Wenn ein neues Objekt erzeugt wird, werden in ihm automatisch das aktuelle Datum und die aktuelle Uhrzeit abgespeichert - allerdings nicht in Textform, sondern als binäre Bitfolgen. Will man Datum und Uhrzeit ausgeben, muss man sie sich also zuerst in Textform konvertieren lassen. Dies macht die Methode toLocaleString().

```
var datum = Datumsobjekt.toLocaleString();
```

Das Ergebnis speichern wir wieder in einer Variablen, deren Wert wir schließlich ausgeben können.

```
document.write(datum);
```

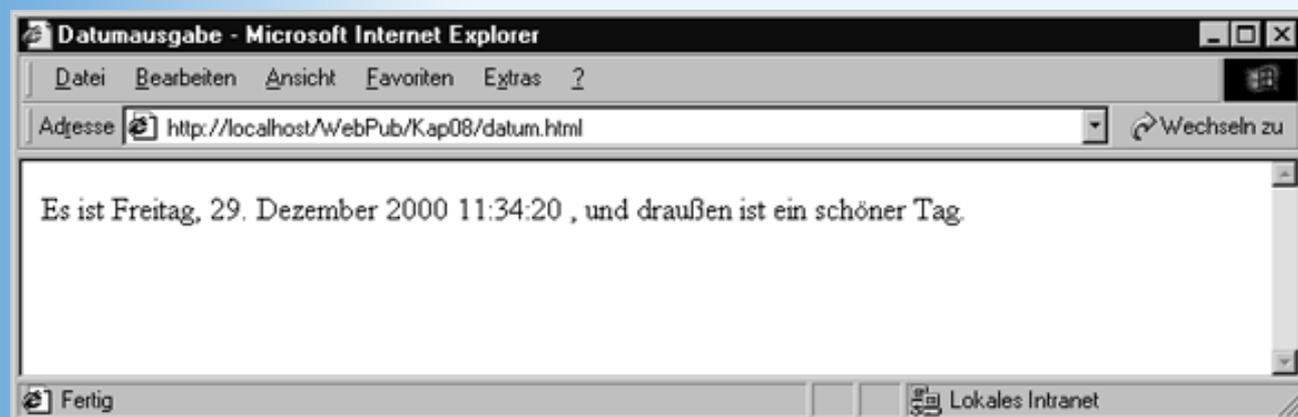


Abbildung 8.5: Aktuelles Datum anzeigen

Meldungsfenster anzeigen

Mit JavaScript können Sie auch Meldungsfenster anzeigen.

Listing 8.9: meldungen.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Meldungen</title>
<script type="text/javascript">
  alert("Hallo!\nDiese Webseite enthält JavaScript-Code.");
</script>
</head>
<body>
<p>Hallo!</p>
</body>
</html>
```

Diese Webseite begrüßt den Webbesucher jedes Mal mit einem Meldungsfenster, dass ihn darauf hinweist, dass man nur mit einem JavaScript-fähigen Browser in den vollständigen Genuss dieser Webseite kommt. Das Beispiel demonstriert noch einmal die Verknüpfung einer JavaScript-Funktion mit einem Ereignis, in diesem Fall das Laden-Ereignis der Webseite (<body>-Tag).

Das Meldungsfenster wird über die Methode alert() aufgerufen. Dieser muss man nur noch den im Meldungsfenster anzuzeigenden Text übergeben. Der anzuzeigende Text wird übrigens automatisch umgebrochen. Möchte man an einer speziellen Stelle selbst einen Zeilenumbruch vorgeben, braucht man nur den Escape-Code \n einzufügen.



Wenn Sie den Text in den Anführungszeichen durch Drücken der (Return)-Taste umbrechen, wird der Umbruch nicht nur bei der Anzeige übergangen, er führt sogar dazu, dass Ihr Skript-Code fehlerhaft ist. Wenn Sie dennoch zur besseren Lesbarkeit des Quellcodes Umbrüche in einen String-Text einfügen wollen, müssen Sie als letztes Zeichen in der Zeile \ eintippen und dann die (Return)-Taste drücken.

Vielleicht wundert es Sie, dass ich hier von der Methode alert() spreche, obwohl es sich doch augenscheinlich um eine Funktion handelt. Schließlich müssen Methoden ja über Objekten aufgerufen werden, und dies ist hier nicht der Fall. Nun, alert() ist tatsächlich eine Methode und das zugehörige Objekt lautet window. Das window-Objekt wird vom Browser eingerichtet und repräsentiert das aktuelle Browser-Fenster, in dem eine Webseite angezeigt wird. Es ist sozusagen, der äußerste Rahmen für all unsere JavaScript-Aktivitäten und nimmt insofern eine Sonderstellung ein, als man in den meisten Fällen auf die explizite Voranstellung des window-Objekts verzichten kann.



So handelt es sich bei dem document-Objekt im Grunde um eine Eigenschaft des window-Objekts und statt document.write müssten wir eigentlich schreiben: window.document.write.

Noch etwas sollte dieses Beispiel demonstrieren. Das Aufrufen von Meldungsfenstern, insbesondere das automatische Aufrufen beim Laden einer Webseite, kann für den Webbesucher recht nervig sein.

Auf Ereignisse reagieren

Die Grundlagen der Ereignisbehandlung in Webseiten, haben Sie bereits in Abschnitt 8.2.2 kennen gelernt. In diesem Abschnitt werden wir diese Technik mit der Anzeige eines Meldungsfensters verbinden.

Die folgende Webseite zeigt ein Bild des Uranus. Klickt der Besucher der Webseite auf das Bild des Uranus erscheint ein Meldungsfenster mit Informationen zu dem Planeten.

Listing 8.10: onclick1.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>JavaScript</title>
  <style type="text/css">
    body { background-color: black }
    h1   { color: white }
    img  { margin: 50px }
  </style>
  <meta http-equiv="Content-Script-Type" content="text/javascript" />
  <script type="text/javascript">
    function hinweis()
    {
      alert("Der gekippte Planet.\n\nMan nimmt an, dass seine Drehachse \
durch einen Zusammenprall mit einem anderen Himmelskörper umgekippt \
wurde.");
    }
  </script>
</head>
<body>
<h1>Der Uranus</h1>

</body>
</html>
```



Abbildung 8.6: Meldungsfenster auf Mausclick

In diesem Beispiel haben wir die Funktion `hinweis()` mit dem `onclick`-Ereignis eines ``-Tags verbunden. Obwohl dies laut HTML 4-Standard absolut korrekt ist, klappt es nicht im Navigator 4, da dieser keine `onclick`-Ereignisse für Bilder unterstützt. Man steht nun vor der Wahl, auf Unterstützung des Navigator 4 zu verzichten (der mittlerweile verfügbare Netscape 6-Browser unterstützt ja das `onclick`-Ereignis, siehe Abbildung 8.6), oder den HTML-Code entsprechend umzuschreiben. Entschließt man sich für Letzteres kann man so vorgehen, dass man

1. das Bild in ein Anker-Element einschließt
2. als Wert für `href` die JavaScript-Funktion angibt (mit `javascript` als »Protokoll«)
3. im ``-Tag das `deprecated`-Attribut `border` auf 0 setzt.

Listing 8.11: Auszug aus `onclick2.html`

```
<a href="javascript:hinweis()">
  
</a>
```

Etwas unschön an diesem Beispiel ist, dass unsere Zusatzinformation in einem Meldungsfenster angezeigt wird, das den Titel »Warnung« trägt oder ein Warnzeichen enthält. Man könnte dies durch Öffnen eines eigenen Browserfenster zum Anzeigen des Textes vermeiden.

Neues Browserfenster öffnen

Mit Hilfe der `open`-Methode des `window`-Objekts kann man ein neues Browserfenster öffnen. Wir wollen dies in nachfolgenden Beispiel dazu nutzen, den Informationstext zum Planeten Uranus (siehe vorangehender Abschnitt) in einem eigenen kleinen Fenster statt in einem Meldungsfenster auszugeben.

Listing 8.12: fenster.html - mit JavaScript ein neues Browserfenster öffnen

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>JavaScript</title>
  <style type="text/css">
    body { background-color: black }
    h1   { color: white }
    img  { margin: 50px }
  </style>
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
  function hinweis()
  {
    var neuesFenster = open("", "Uranus", "height=150,width=200");
    neuesFenster.document.open();

    neuesFenster.document.write("<html><head>");
    neuesFenster.document.write("<title>Uranus</title></head>");
    neuesFenster.document.write("<body><p>Man nimmt an, dass seine \
      -Drehachse durch einen Zusammenprall mit einem anderen \
      -Himmelskörper umgekippt wurde.</p>");
    neuesFenster.document.write("</body></html>");
    neuesFenster.document.close();
  }
</script>
</head>
<body>
<h1>Der Uranus</h1>

</body>
</html>
```

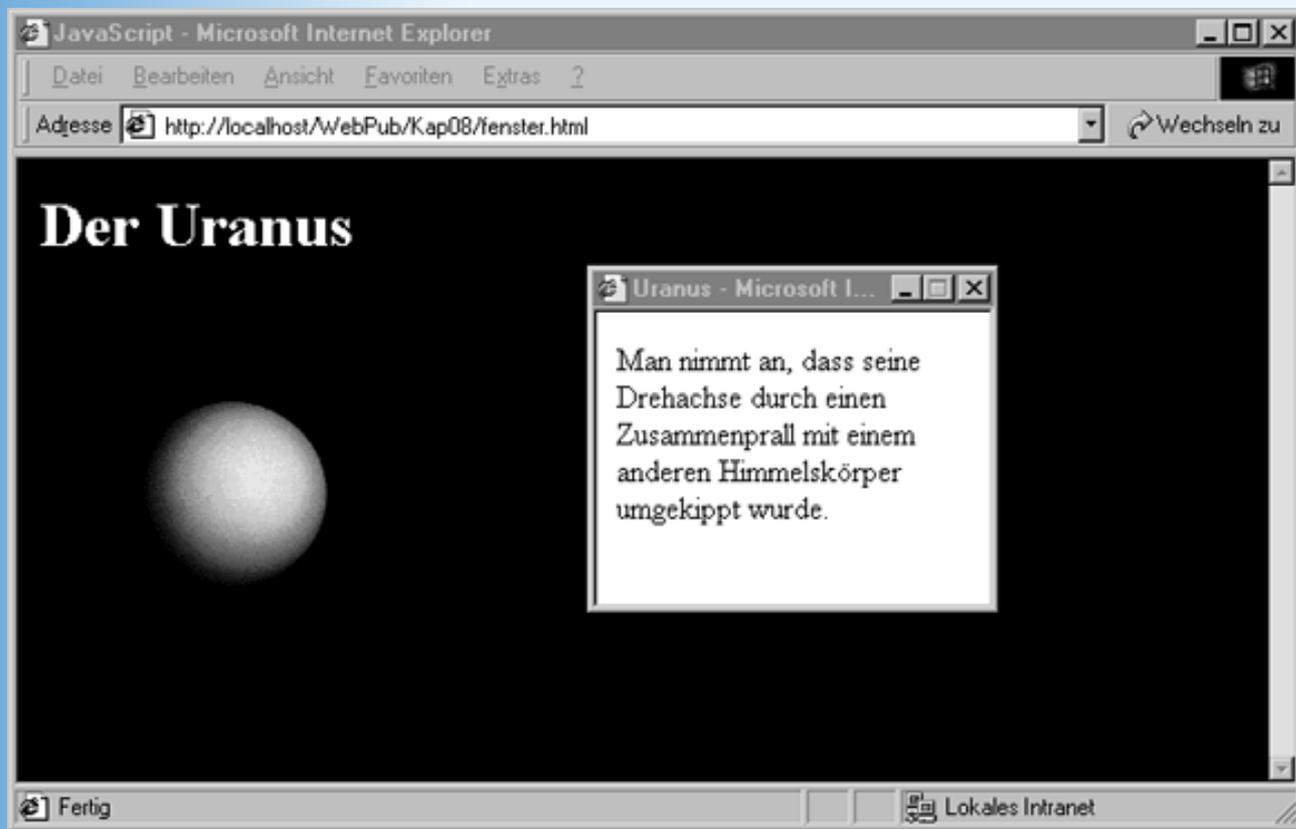


Abbildung 8.7: Mit JavaScript generiertes Browserfenster

Erzeugt wird das neue Browserfenster mit Hilfe der Methode `window.open`:

```
var neuesFenster = open("", "Uranus", "height=150,width=200");
```



Das Objekt `window` muss wie gesagt nicht explizit angegeben werden, da der Browser es automatisch ergänzt.

Das erste Argument zu der Methode ist der URL der Webseite, die in dem Browserfenster angezeigt werden soll. Da wir den HTML-Code der Webseite nachfolgend im JavaScript-Code generieren werden, brauchen wir keinen URL anzugeben. Das zweite Argument ist der Name des Browserfensters. Dieser Name kann beispielsweise zusammen mit dem `target`-Attribut von Hyperlinks verwendet werden (siehe Kapitel 3.2.4). Beachten Sie, dass dieser Name keine Leerzeichen enthalten sollte, da diese vom Internet Explorer nicht korrekt verarbeitet werden. Das letzte Argument ist eine Liste von Fenstereigenschaften. Wir beschränken uns an dieser Stelle darauf, die Höhe und Breite des Fensters vorzugeben.

Danach erzeugen wir in dem Browserfenster eine neue Webseite...

```
neuesFenster.document.open();
```

... und geben den HTML-Code der Webseite aus:

```
neuesFenster.document.write("<title>Uranus</title></head>");
```

```
neuesFenster.document.write("<body><p>Man nimmt an, dass seine \
    -Drehachse durch einen Zusammenprall mit einem anderen \
    -Himmelskörper umgekippt wurde.</p>");
neuesFenster.document.write("</body></html>");
```

Zum guten Schluss rufen wir die close-Methode des document-Objekts auf, wodurch dem Browser mitgeteilt wird, dass die Webseite fertig ist und angezeigt werden kann.

```
neuesFenster.document.close();
```

In die Statusleiste des Browsers schreiben

Man kann mit JavaScript auch in die Statusleiste des Browsers schreiben.

```
window.status="Dieser Text erscheint in der Statusleiste des Browsers."
```

Von dieser Technik sollte man allerdings möglichst wenig Gebrauch machen, überlassen Sie die Statusleiste dem Browser.

Das Erscheinungsbild einer Webseite ändern

Mit JavaScript können Sie auch auf die Gestaltung oder den Aufbau einer Webseite Einfluss nehmen.

Als einfaches Beispiel dazu haben Sie bereits gesehen, wie man über die Eigenschaft bgcolor des document-Objekts den Hintergrund einer Webseite einfärben kann.

Listing 8.13: hintergrundfarbe.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <title>JavaScript</title>
    <meta http-equiv="Content-Script-Type" content="text/javascript" />
</head>
<body onload="document.bgColor='cyan';">
<h1>JavaScript</h1>
<p>Wenn Sie eine schwarze Schrift vor zyanblauem Hintergrund sehen, wurde der
JavaScript-Code korrekt ausgeführt.</p>
</body>
</html>
```

Das Erscheinungsbild einer Webseite kann aber nicht nur über die Eigenschaften des document-Objekts verändert werden. Mit JavaScript können Sie auch auf die Attribute und Stylesheet-Eigenschaften der einzelnen HTML-Elemente zugreifen. Die Art und Weise, in der man auf die HTML-Elemente und ihre Attribute und Stileigenschaften zugreifen kann, ist allerdings etwas komplizierter und - leider - von Browser zu Browser verschieden. Wir werden uns in Kapitel 10 ausführlicher mit diesem Thema befassen.

Formulare bearbeiten und Benutzeroberflächen aufbauen

Eine der Haupteinsatzgebiete für JavaScript ist die clientseitige Bearbeitung von Formulareingaben. So kann man beispielsweise mit Hilfe von JavaScript sicherstellen, dass ein Formular vollständig und korrekt ausgefüllt wurde, bevor man es an ein serverseitiges CGI-Programm weiterreicht (siehe Kapitel 11).

Man kann aber auch Benutzeroberflächen implementieren, über die der Websurfer mit einer Webseite interagieren kann. Als kleines Beispiel zeigen wir Ihnen hierzu eine Webseite deren Hintergrundfarbe vom Besucher der Webseite eingestellt werden kann.

Listing 8.14: optionsfelder.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>JavaScript</title>
  <meta http-equiv="Content-Script-Type" content="text/javascript" />
</head>
<body text="maroon" style="color: red">
<h1>JavaScript</h1>
<form>
<input type="radio" name="R1" value="V1"
  onclick="document.bgColor='white'">Weiß
<input type="radio" name="R1" value="V2"
  onclick="document.bgColor='blue'">Blau
<input type="radio" name="R1" value="V3"
  onclick="document.bgColor='black'">Schwarz</p>
</form>
<p>Nutzen Sie obige Optionsfelder um eine Ihnen zusagende Hintergrundfarbe
einzustellen.</p>
</body>
</html>
```

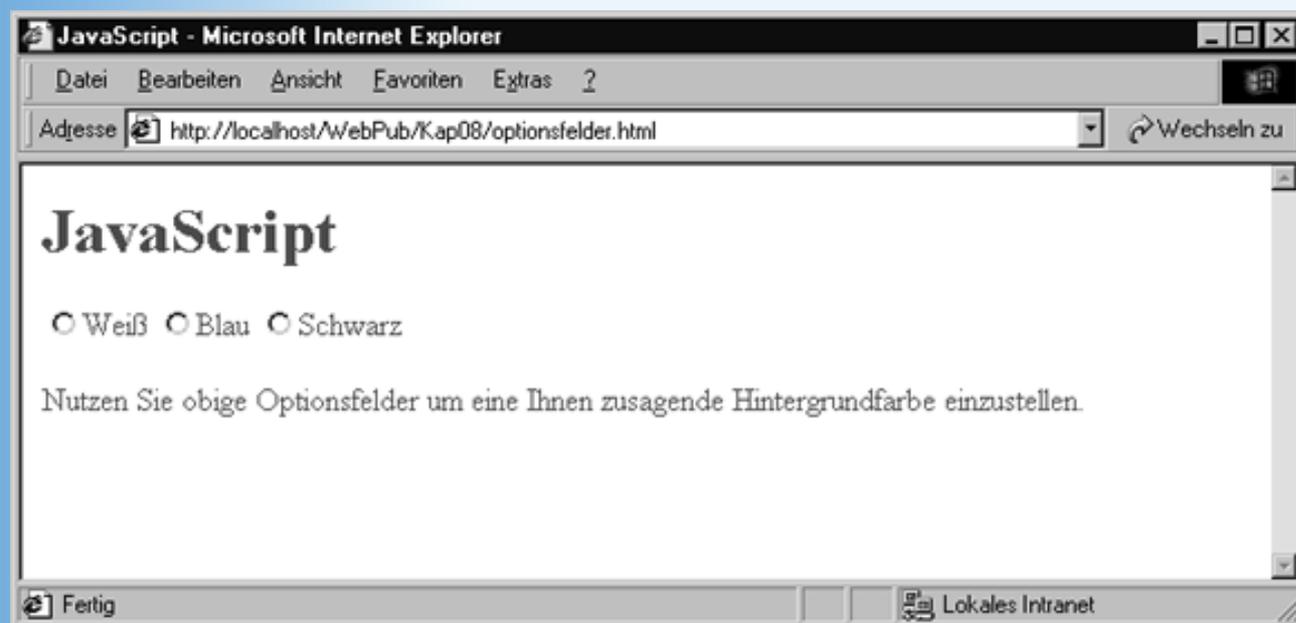


Abbildung 8.8: Formulare als aktive Elemente der Benutzeroberfläche



Wenn Ihnen die Arbeit mit Formularen und Steuerelementen nicht mehr gegenwärtig ist, lesen Sie noch einmal in Kapitel 3 nach. Mehr zur Programmierung von Benutzeroberflächen mit Hilfe von JavaScript und Formularen/Steuerelementen erfahren Sie in Kapitel 11.

DHTML und Animationen

Last but not least können Sie mit JavaScript auch HTML-Elemente frei bewegen und Animationen implementieren. Wie dies geht, erfahren Sie in Kapitel 16.3.



Der Begriff DHTML, dynamisches HTML, wird in der Literatur unterschiedlich weit gefasst. Im engeren Sinne steht DHTML für die dynamische Änderung der Sichtbarkeit und Positionierung von HTML-Elementen. Im weiteren Sinne steht DHTML für jede Art von JavaScript-Zugriff auf HTML-Elemente. Im Laufe dieses Buches werden wir beide Formen von DHTML erläutern und Beispiele dazu zeigen.

8.4 Fehler im JavaScript-Code finden

Egal, ob Sie in die Webseitenprogrammierung mit JavaScript neu einsteigen oder schon ein gewiefter JavaScript-Fuchs sind, Fehler werden Sie immer machen. Nun, das ist nicht weiter schlimm, man muss nur wissen, wie man die Fehler ausfindig machen und beheben kann.

Grundsätzlich ist dabei zwischen zwei Arten von Fehlern zu unterscheiden:

- Syntaxfehlern und
- Logischen Fehlern

Syntaxfehler

Ein Syntaxfehler liegt vor, wenn der von Ihnen aufgesetzte JavaScript-Code nicht den JavaScript-Syntaxregeln entspricht und daher vom Interpreter nicht verstanden wird. Solche Fehler entstehen meist durch Unwissenheit (»hieß die Eigenschaft nun `hgColor` oder `bgColor`?«), durch Vergesslichkeit (Sie haben vergessen, eine JavaScript-Funktion mit einer geschweiften Klammern abzuschließen) oder durch Tippfehler (Sie haben `bgcolor` statt `bgColor` eingetippt).

Manchmal braucht man das Skript nur noch einmal sorgsam anzuschauen, um den Fehler zu entdecken. Es gibt aber auch Fälle, in denen man förmlich wie der Ochse vorm Berg steht und den Fehler selbst beim hundertsten Drüberlesen nicht sieht, oder eine Webseite enthält einfach viel zu viel Code, als dass man diesen aufmerksam Zeile für Zeile durchgehen wollte. Betrachten Sie doch einmal das Listing 8.15. Sehen Sie, wo der Fehler liegt?

Listing 8.15: debuggen.html - Webseite mit Fehler im Skriptcode

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Datumausgabe</title>
<script type="text/javascript">
  alert("Hallo!
      Diese Webseite enthält JavaScript-Code.");
</script>
</head>
<body>
<p>Hallo!</p>
</body>
</html>
```

Sie meinen, dass das Problem mit dem Zeilenumbruch in dem Text der alert-Methode zu tun hat? Absolut richtig. JavaScript erlaubt keine Umbrüche in String-Konstanten (Text in Anführungszeichen). Um den Fehler zu beheben, könnten Sie den Code auf eine der folgenden Weisen ändern:

- Umbruch entfernen

```
alert("Hallo! Diese Webseite enthält JavaScript-Code.");
```

- Umbruch in Quellcode vor JavaScript-Interpreter verstecken

```
alert("Hallo! \
Diese Webseite enthält JavaScript-Code.");
```

- Umbruch in Quellcode vor JavaScript-Interpreter verstecken und Umbruch in Meldungsfenster ausgeben

```
alert("Hallo!\n \
Diese Webseite enthält JavaScript-Code.");
```

Gut, nun haben wir noch einmal das Thema Zeilenumbrüche in JavaScript-Stringkonstanten aufgefrischt. Was uns aber in diesem Kapitel vor allem interessiert, ist, wie uns der Browser beim Auffinden solcher Syntaxfehler unterstützen kann.

Webseiten im Browser testen

Wenn Sie obige, fehlerbehaftete Webseite in einen Browser laden, wird der JavaScript-Interpreter über den fehlerhaften Umbruch stolpern. Wie er danach weiter verfährt, ist von Browser zu Browser unterschiedlich. Der Netscape 6-Browser zeigt beispielsweise Hallo als Inhalt der Webseite an und bricht dann ab (siehe Abbildung 8.9).

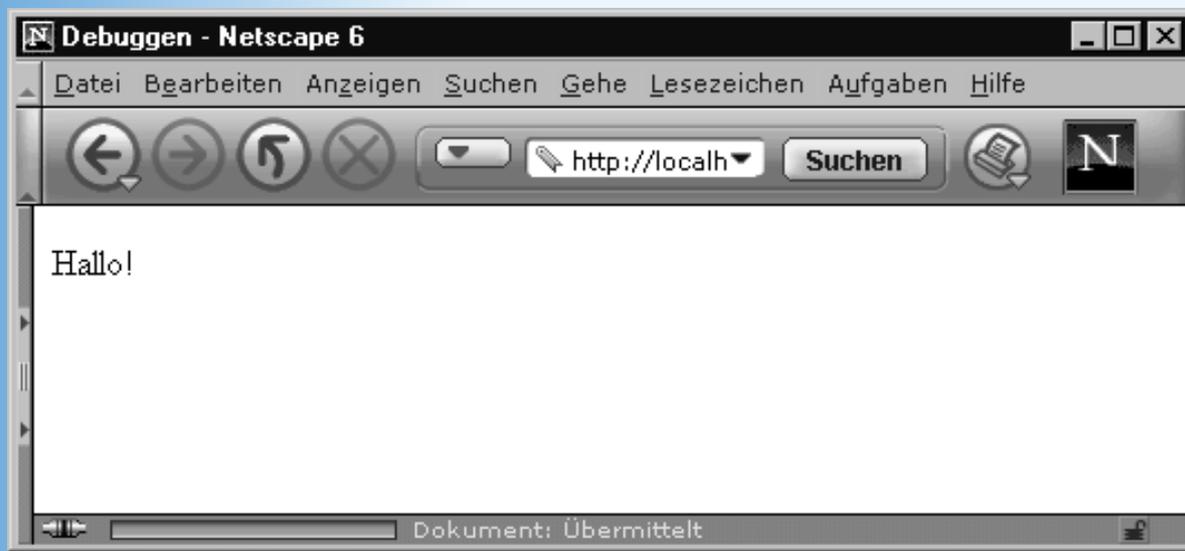


Abbildung 8.9: Fehlerhaft dargestellte Webseite

Man kann sich aber in den meisten Browsern Hinweise auf die mögliche Fehlerquelle anzeigen lassen.



Beachten Sie, dass der Browser nur Hinweise auf die mögliche Fehlerursache geben kann. Manchmal führt Sie ein solcher Hinweis direkt zu dem wirklichen Fehler, manchmal ist der Fehler schon etwas weiter oben passiert und der Browser hat es nur etwas später bemerkt.

Internet Explorer

Damit Sie der Internet Explorer auf die mögliche Quelle von Syntaxfehlern in Ihrem JavaScript-Code hinweist, müssen Sie die Option **Skriptfehler anzeigen** aktivieren. Rufen Sie dazu den Befehl **Extras/Internetoptionen** auf und markieren Sie auf der Registerkarte **Erweitert** die Option (siehe Abbildung 8.10).

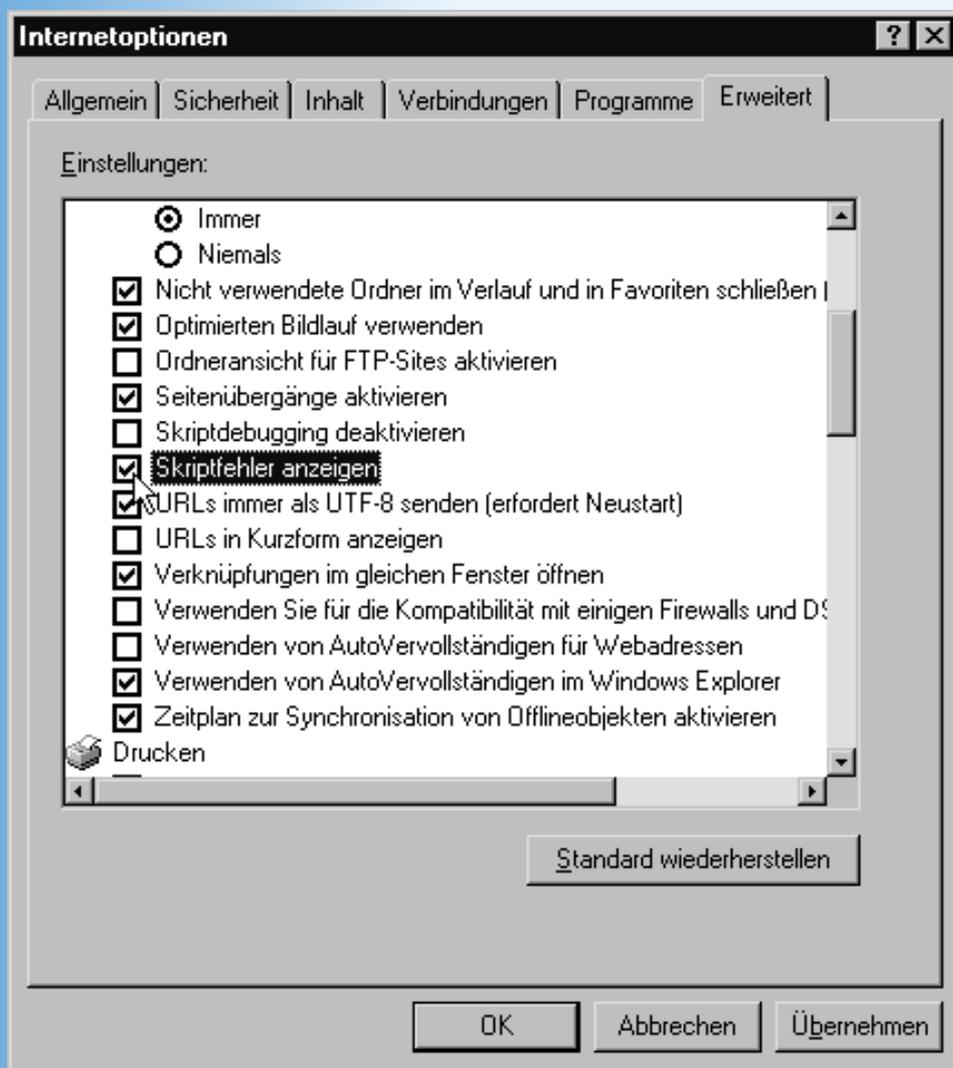


Abbildung 8.10: Anzeige von Skriptfehlern aktivieren

Tritt danach beim Ausführen eines Skripts ein Fehler auf, stoppt der Browser die Ausführung des Skripts und zeigt Ihnen ein Meldungsfenster⁴ mit einem Hinweis auf die mögliche Fehlerquelle an.

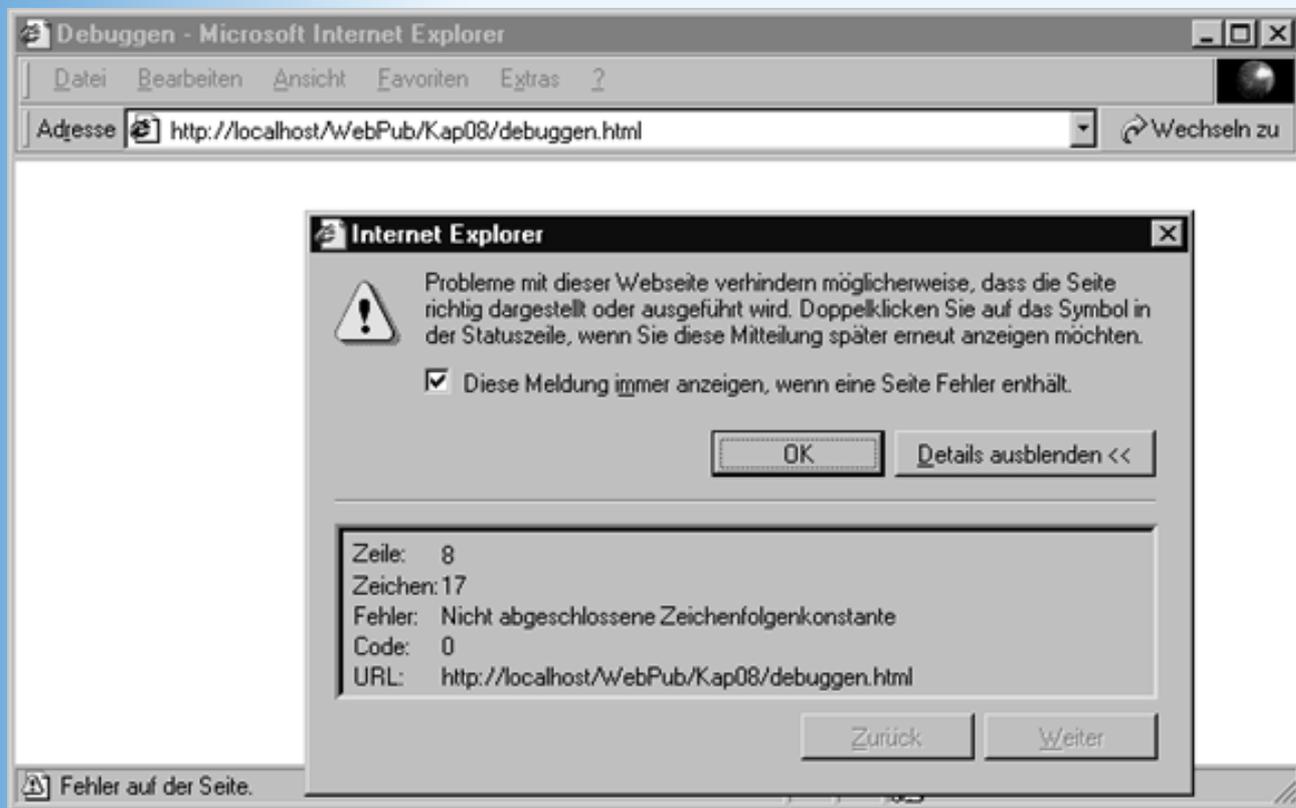


Abbildung 8.11: Hinweis auf Skriptfehler im Internet Explorer

Netscape Communicator 4

Im Navigator 4 müssen Sie nach dem Laden der Webseite im Adressenfeld

javascript:

eingeben und die (Return)-Taste betätigen. Der Browser ruft daraufhin die Communicator-Konsole auf, in der Hinweise auf die mögliche Fehlerursache angezeigt werden.

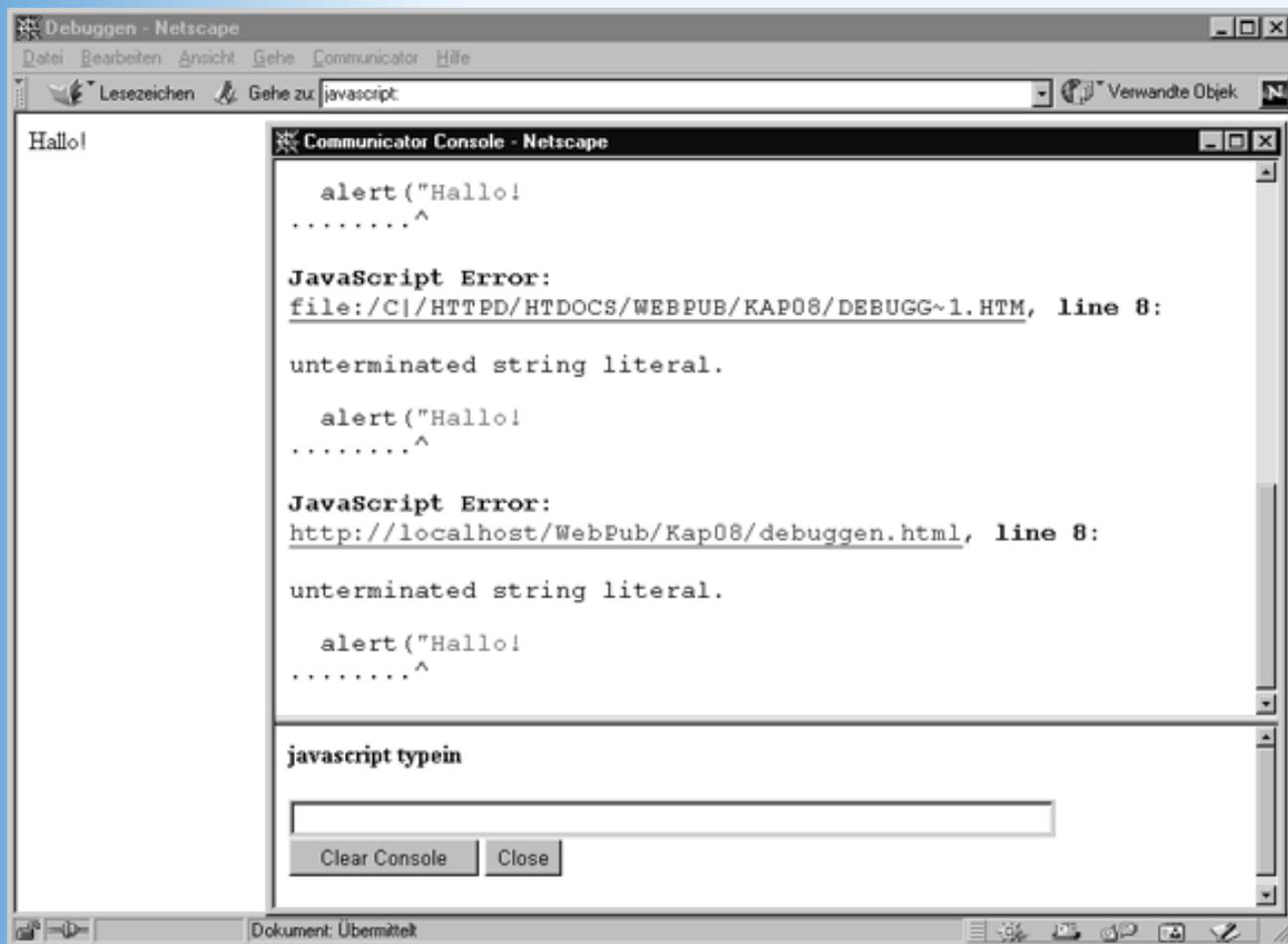


Abbildung 8.12: Hinweis auf Skriptfehler im Netscape Navigator

Netscape 6-Browser

Im Netscape 6-Browser rufen Sie nach dem Laden der Webseite den Menübefehl **Aufgaben/Extras/JavaScript-Konsole** auf. In dieser können Sie sich dann über die möglichen Fehler informieren.



Abbildung 8.13: Hinweis auf Skriptfehler im Netscape 6-Browser



*In der Konsole werden unter Umständen auch ältere Fehler angezeigt. Achten Sie nur auf die letzten Fehler oder **Löschen** Sie die Anzeige, wechseln Sie zurück in den Browser und lassen Sie die Webseite neu laden.*

Logische Fehler

Logische Fehler liegen vor, wenn der Skript-Code syntaktisch korrekt ist, aber bei der Ausführung (Programmierer sagen »zur Laufzeit«) nicht das macht, was wir von ihm erwarten. Bei solchen Fehlern kann der JavaScript-Interpreter nicht helfen. Es bleibt uns daher nichts anderes übrig als selbst zu analysieren, was schief läuft und wo der Fehler liegen könnte.

Helfen kann Ihnen dabei ein Debugger. Ein Debugger ist ein spezielles Programm zur Lokalisierung von Laufzeitfehlern. Mit seiner Hilfe kann man den Code eines Programms Schritt für Schritt ausführen lassen und dabei die Ausgabe und die Werte der Variablen des Programms verfolgen.

Für die meisten Browser gibt es passende JavaScript-Debugger, die man bei der Installation mit einrichten lassen kann (Netscape 4: <http://developer.netscape.com:80/software/jsdebug.html>; IE: <http://msdn.microsoft.com/scripting/debugger/default.htm>).

8.5 Zusammenfassung

Heute haben wir unsere ersten JavaScript-Programme zur Unterstützung von Webseiten geschrieben. JavaScript ist eine einfache und dennoch mittlerweile äußerst leistungsfähige Sprache zur clientseitigen Programmierung.

JavaScript-Code kann man mit Hilfe von `<script>`-Tags und/oder einem Ereignisattribut direkt in eine Webseite einfügen oder auch aus einer Datei laden.

```
<script type="text/javascript" src="farben.js"></script>
```

Wichtig ist, dass man nie vergisst, dem Browser die verwendete Skriptsprache anzuzeigen, damit er weiß, welchen integrierten Interpreter er aufrufen muss. Abgesehen davon, dass man für jedes `<script>`-Tag eine eigene Skriptsprache angeben kann, ist es auch möglich im Header eine Standardskriptsprache vorzugeben:

```
<meta http-equiv="Content-Script-Type" content="text/javascript" />
```

Nach den Formalitäten haben wir ein wenig in die Praxis hinein geschnuppert und uns verschiedene Techniken und Einsatzmöglichkeiten für JavaScript angeschaut:

- Dynamischen Text ausgeben
- Aktuelles Datum ausgeben
- Meldungsfenster anzeigen
- Auf Ereignisse reagieren
- Neues Browserfenster öffnen
- In die Statusleiste des Browsers schreiben
- Das Erscheinungsbild einer Webseite ändern
- Formulare bearbeiten und Benutzeroberflächen aufbauen
- Animationen

Dabei haben Sie auch schon ein wenig über Objekte, Klassen und Strings in JavaScript erfahren.

Zum Schluss haben wir uns angeschaut, wie man sich vom Browser Hinweise auf syntaktische Fehler im Skriptcode anzeigen lassen kann.

8.6 Fragen und Antworten

Frage:

Welche anderen clientseitigen Skriptsprachen gibt es?

Antwort:

Neben JavaScript sind die wichtigsten Skriptsprachen tcl (unter UNIX/Linux) und VBScript (Microsoft-Plattform), denen allerdings bei weitem nicht die Bedeutung und breite Unterstützung seitens der großen Browser zuteil wird, die JavaScript erfährt.

Frage:

Welche Beziehung besteht zwischen Java und JavaScript?

Antwort:

Praktisch keine, obwohl man mit beiden Sprachen Programmcode für Webseiten aufsetzen kann, der clientseitig ausgeführt wird. Doch damit hören die Gemeinsamkeiten fast schon auf.

- Java wurde von Sun entwickelt, JavaScript von Netscape.
- Beides sind also völlig unabhängige Produkte. Natürlich gibt es gewisse Ähnlichkeiten in der Syntax beider Sprachen, doch sind diese ziemlich rudimentär und nur auf der untersten Ebene und im groben Aufbau zu finden (Bezeichnung für Operatoren, Steuerung des Programmablaufs, vergleichbare objektorientierte Konzepte etc.).
- Java-Programme liegen stets als eigene, vorkompilierte Programmdateien vor, während JavaScript-Skripte immer als reiner ASCII-Text vorliegen, der direkt im HTML-Code der Webseiten oder einer externen Textdatei steht.
- Darüber hinaus ist Java eine vollwertige Programmiersprache, mit der man nicht nur Applets zur Unterstützung von Webseiten, sondern auch ganz normale Programme entwickeln kann, während JavaScript ganz auf die Erweiterung von Webseiten zugeschnitten ist.

Frage:

Serverseitige Programmierung bedeutet stets ein gewisses Sicherheitsrisiko für den Server-Rechner. Die clientseitige Programmierung mit JavaScript verschont den Webserver, aber stellt sie nicht ein Risiko für die Client-Rechner dar, also die Rechner der Websurfer, die sich die mit JavaScript versehenen Webseiten herunterladen?

Antwort:

Ja und nein! Natürlich stellt die Ausführung eines fremden Programmcodes (und nichts anderes ist JavaScript-Code) ein Sicherheitsrisiko für den Client-Rechner dar. Andererseits ist die Sprache so konzipiert, dass man mit JavaScript keinen Code aufsetzen kann, der dem Client-Rechner schaden könnte. So ist es üblicherweise nicht möglich, von dem JavaScript-Code einer Webseite auf den Inhalt von Webseiten eines anderen Webs zuzugreifen, die in einem zweiten Browserfenster (Websurfer surfen nicht selten parallel mit mehreren Browsern) angezeigt werden. Genauso wenig ist es JavaScript-Code erlaubt, auf die Festplatte des Client-Rechners zuzugreifen (das größte Sicherheitsrisiko überhaupt). JavaScript-Skripte die dies versuchen, müssen sich zuvor eine Erlaubnis beim Websurfer einholen (durch Dialogfenster oder Nachweis einer Authentifizierung (signierte Skripte)). Wir werden uns mit diesen Techniken allerdings nicht weiter beschäftigen, da wir der Meinung sind, dass man sie am besten gar nicht nutzen sollte. Kontrolliert werden diese Sicherheitsmaßnahmen letztendlich nur durch

den Browser und seinen JavaScript-Interpreter. Wenn Sie also Vertrauen in Ihren Browser haben, können Sie auch Vertrauen in den JavaScript-Code haben. ;-)

Frage:

Ich habe Probleme mit der Nachprogrammierung der Beispiele aus diesem Kapitel. Woran könnte dies liegen?

Antwort:

Es kann hierfür natürlich eine Reihe von Gründen geben. Meist werden Sie sich einfach irgendwo vertippt haben. Oder Sie unterscheiden nicht zwischen Groß- und Kleinschreibung. Denken Sie daran: Für den JavaScript-Interpreter ist `bgColor` korrekt, `bgcolor` nicht. Wenn Sie den Fehler gar nicht entdecken können, laden Sie die Webseiten von der Buch-CD.

8.7 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

Quiz

1. Was ist der Unterschied zwischen clientseitiger und serverseitiger Programmierung?
2. Was ist ein Interpreter?
3. Wie bettet man Skriptcode in eine Webseite ein?
4. Wie lädt man Skriptcode aus einer Skriptdatei?
5. Welche Extension haben JavaScript-Dateien?
6. Wie lautet das JavaScript-Objekt, das eine Webseite repräsentiert?
7. Wie lautet das JavaScript-Objekt, welches das Browserfenster repräsentiert?

Übungen

1. Suchen Sie sich ein Beispiel aus diesem Kapitel aus, programmieren Sie es nach und testen Sie es in Ihrem Browser. Wenn alles funktioniert, bauen Sie zwei, drei Fehler in den JavaScript-Code ein und spüren Sie die Fehler mit Hilfe Ihres Browsers auf.

1

Der Einfachheit halber werden wir in der Folge weiter von JavaScript statt von ECMA-Skript sprechen.

2

Eine Methode ist eine Funktion, die zu einer bestimmten Klasse von Objekten gehört und nur über ein Objekt aufgerufen werden kann. Mehr dazu in Kapitel 9.

3

Über welche Eigenschaften ein Objekt verfügt, wird von der Klasse vorgegeben (das heißt, alle Objekte einer Klasse haben die gleichen Eigenschaften (und Methoden)). Welche Werte die Eigenschaften eines Objekts haben, ist dagegen objektspezifisch.

4

Das Meldungsfenster des Internet Explorers kann unterschiedlich aufgebaut sein.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

Tag 9

JavaScript- Grundkurs

Wie man JavaScript einsetzt und was man so mit JavaScript anfangen kann, wissen Sie jetzt. Um aber richtig mit JavaScript programmieren zu können, um nicht bei jeder noch so kleinen JavaScript-Funktion mit Syntaxfehlern kämpfen zu müssen und um anspruchsvollere JavaScript-Funktionen programmieren zu können, müssen Sie mehr über die grundlegende JavaScript-Syntax lernen.

In dieser Hinsicht ist JavaScript nichts anderes als eine ganz normale höhere Programmiersprache, und viele Konzepte, die Sie heute am Beispiel von JavaScript kennen lernen werden, kann man auch in anderen Programmiersprachen wiederfinden. Leser, für die dieses Kapitel der Einstieg in die Programmierung ist, werden von dem hier Gelernten also auch beim Erlernen weiterer Programmierkapitel profitieren (wir denken hier vor allem an die dritte Woche). Umgekehrt werden sich Lesern, die bereits über Programmiererfahrung verfügen, schnell und ohne Probleme in die Programmierung mit JavaScript einarbeiten.

Die Themen im Einzelnen:

- Wie werden Daten in Programmen repräsentiert?
- Wie kann man Daten verarbeiten?
- Wie kann man den Programmablauf steuern?
- Wie definiert man Funktionen?
- Wie kann man mit einer Funktion Daten austauschen?
- Was sind Objekte? Was ist objektorientierte Programmierung?
- Welche vordefinierten Klassen und Objekte gibt es?

9.1 Variablen, Konstanten und Datentypen

Programme dienen der Verarbeitung von Daten, wobei es sich ebenso gut um Zahlen (3 oder 54) wie um Strings ("`<p>Hallo JavaScript</p>`") oder die Hintergrundfarbe einer Webseite handeln kann. Um aber sinnvoll mit Daten programmieren zu können, muss es eine Möglichkeit geben, wie man Daten in einem Programm zwischenspeichern kann.

Denken Sie nur daran, wie Sie eine Folge von Zahlen, beispielsweise 3 - 15 - 7 - 4 addieren. Sie addieren 3 und 15 behalten das Ergebnis im Kopf und addieren dann 7, speichern das Ergebnis wieder im Kopf, addieren 4 und haben im Kopf dann das Ergebnis (29, wenn mein Kopfrechnen mich nicht im Stich gelassen hat).

Variablen repräsentieren Daten

Wo aber kann der Rechner Werte zwischenspeichern? Richtig, im Arbeitsspeicher. Der Arbeitsspeicher besteht aus einer Folge von einzelnen Speicherzellen, die durchgehend nummeriert sind. Dies sind die sogenannten Speicheradressen. Will man einen Wert im Speicher ablegen, sucht man sich einen genügend großen Speicherplatz und merkt sich die Anfangsadresse sowie die Größe des Speicherplatzes. Das klingt kompliziert und es ist auch kompliziert. Deshalb nimmt uns der Interpreter die gesamte komplizierte Arbeit der Speicherverwaltung ab. Alles, was wir tun müssen, wenn wir Speicher zur Aufbewahrung von Daten benötigen, ist, uns Variablennamen auszudenken und diesen die Daten zuzuweisen.



Wenn Sie eine Variable definieren, reserviert der Compiler für diese Variable Speicher im Arbeitsspeicher des Computers und verbindet den Variablennamen mit diesem Speicher. Danach können Sie der Variablen Werte zuweisen oder den Wert der Variablen abfragen.

Wie könnte demnach die obige Aufsummierung als JavaScript-Code aussehen?

Listing 9.1: Auszug aus variablen.html

```
<body>
<h1>Variablen</h1>
<script type="text/javascript">
  var summe = 3;
  summe = summe + 15;
  summe = summe + 7;
  summe = summe + 4;
  document.write("<p>Die Summe von 3, 15, 7 und 4 lautet : ");
  document.write(summe);
  document.write("</p>");
</script>
</body>
```

Variablen definieren

In diesem Beispiel haben wir unsere Variable `summe` genannt. Variablennamen sind - unter Einhaltung einiger weniger Regeln, siehe unten - frei wählbar, sollten aber möglichst so gewählt werden, dass man aus dem Namen auf die Verwendung/den Inhalt der Variablen schließen kann.

Wenn wir eine Variable definieren, stellen wir dem Variablennamen das Schlüsselwort `var` voran - dies zeigt uns und dem Interpreter an, dass hier eine Variablendefinition folgt. Schließlich weisen wir der Variablen noch im Zuge der Definition einen anfänglichen Wert (im Beispiel 3) zu.



Die Zuweisung eines Anfangswerts ist nicht zwingend erforderlich, aber empfehlenswert (außer der Variablen wird kurz nach der Definition ein Anfangswert zugewiesen). Wenn Sie keinen sinnvollen Anfangswert für eine Variable wissen, weisen Sie ihr den symbolischen Wert `null` zu.

Übrigens: Wenn Sie mehrere Variablen definieren wollen, müssen Sie das `var`- Schlüsselwort nicht jedes Mal wiederholen. Sie können die zu definierenden Variablen auch durch Kommata getrennt hintereinander auflisten:

```
var zahl1, zahl2, zahl3;
```

Werte zuweisen und abfragen

Danach werden dem Wert der Variablen nacheinander die einzelnen Zahlenwerte hinzuaddiert. Eine Zuweisung wie

```
summe = summe + 15;
```

bedeutet für den Interpreter, dass er zuerst den Ausdruck auf der rechten Seite des Gleichheitszeichen berechnet und dann das Ergebnis in der Variablen auf der linken Seite speichert. Dabei repräsentiert die Variable `summe` auf der linken Seite des Gleichheitszeichens den in ihr gespeicherten Wert. Der Interpreter nimmt also den augenblicklich noch in `summe` gespeicherten Wert 3, addiert zu diesem den Wert 15 und speichert das Ergebnis (18) in `summe`.



Das Zeichen `=` steht in JavaScript nicht für einen Vergleich auf Gleichheit, sondern für eine Zuweisung!

Werte von Variablen ausgeben

Zum Schluss wird das Ergebnis der Berechnung ausgegeben. Den Wert einer Variablen gibt man aus, indem man einfach den Variablennamen als Argument an die write- Methode übergibt.

```
document.write(summe);
```

Regeln für die Namensgebung

In der Wahl Ihrer Variablennamen (wie auch anderer Bezeichner, beispielsweise Funktionen- oder Klassennamen, siehe nachfolgende Kapitel) sind Sie gänzlich frei, solange Sie sich an folgende Regeln halten:

- Erlaubt sind Groß- und Kleinbuchstaben, die Zahlen 0 - 9 und der Unterstrich (_).
- Nicht erlaubt sind Leerzeichen und Punkte.
- Das erste Zeichen muss ein Buchstabe oder ein Unterstrich sein.
- Sie dürfen keines der in JavaScript definierten Schlüsselwörter als Bezeichner verwenden.

break	else	new	var
case	finally	return	void
catch	for	switch	while
continue	function	this	with
default	if	throw	
delete	in	try	
do	instanceof	typeof	

Tabelle 9.1: Schlüsselwörter von JavaScript

Außerdem sollten Sie beachten, dass JavaScript streng zwischen Groß- und Kleinschreibung (was bedeutet, dass summe und Summe zwei vollkommen unabhängige Variablen bezeichnen) und dass man Variablennamen möglichst so wählen sollte, dass man aus dem Namen auf die Verwendung/den Inhalt der Variablen schließen kann.

Konstanten



Werte, die man direkt in den Programmcode schreibt (beispielsweise um sie Variablen zuzuweisen), nennt man Konstanten (oder Literale).

```
var meineZahl, meinString;  
meineZahl = 133;  
meinString = "Hallo";
```

Obiger Code verwendet eine Ganzzahlkonstante (133) und eine Stringkonstante ("Hallo").



Text oder Zeichenfolgen in Programmen bezeichnet man gemeinhin als Strings.

Wie eine Konstante aussieht, hängt von ihrem Datentyp ab.

Datentyp	Literale	Beispiele

Null-Wert	null	var = null;
Boolesche Werte	true oder false	var = true;
Zahlen	Dezimalzahlen: 123, -12 Hexadezimalzahlen (beginnen mit 0x) Fließkommazahlen (Bruchzahlen mit Punkt oder Zahlen in Exponentialschreibweise)	var = 123; var = 0x10; // gleich 16 var = 3.14; var = 1.5e3; // gleich 1500
Strings	Zeichenfolgen, die in einfachen oder doppelten Anführungszeichen stehen.	var = 'Hallo'; var = "Hallo";

Tabelle 9.2: Konstanten

Werte für Variablen vom Websurfer abfragen

Programme, die ausschließlich Daten verarbeiten, die vom Programmierer vorgegeben werden, sind meist recht langweilig. Interessanter wird es, wenn ein Programm Daten verarbeitet, die von außen kommen: beispielsweise die Eigenschaften einer Webseite oder Werte, die der Besucher der Webseite eingibt.

Der übliche Weg, Daten vom Websurfer abzufragen, führt über die Einrichtung einer Benutzeroberfläche mit Steuerelementen und Formularen (siehe Kapitel 11). Manchmal kommt man aber auch mit der vordefinierten window-Methode prompt aus.

```
eingabe = prompt("Aufforderung", "Vorgabe");
```

Als Argumente übergibt man einen Text, in dem man dem Websurfer anzeigt, was für eine Eingabe man vom ihm erwartet, und einen Text, der als Standardeingabe vorgegeben wird (und meist aus einem leeren String "" besteht). Der vom Websurfer eingegebene Wert wird als Ergebnis der prompt-Methode zurückgeliefert und kann in einer Variablen gespeichert werden.

Das nachfolgende Beispiel fordert den Websurfer auf einen DM-Wert einzugeben und rechnet diesen in Euro um. Das Ergebnis der Umrechnung wird danach als HTML-Text ausgegeben.

Listing 9.2: euro.html - Benutzereingaben mit prompt abfragen

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>DM/Euro</title>
</head>
<body>
<h1>Umrechnung von DM in Euro</h1>
<script type="text/javascript">

  var dm_wert    = null;
  var euro_wert  = null;

  dm_wert = prompt("Geben Sie einen Wert in DM an: ", "");
  euro_wert = dm_wert / 1.95583;
  document.write("<p>");
  document.write(dm_wert);
  document.write(" DM entsprechen ");
  document.write(euro_wert);
  document.write(" Euro.</p>");
</script>
</body>
```

</html>



Abbildung 9.1: Die prompt-Benutzereingabe

Fallstricke und Vertiefung

Im Grunde ist die Programmierung mit Variablen die Einfachheit selbst. Doch manchmal hakt es auch und ohne tiefere Kenntnisse der mit den Variablen verbundenen Konzepte sind solche Fehler nicht leicht zu finden.

L- und R-Wert

Wie Sie wissen, sind Variablen mit Speicherzellen verbunden. Über den Variablennamen können Sie sowohl einen Wert in einer Variablen abspeichern als auch den Wert einer Variablen abfragen. Woher aber weiß der Interpreter, ob Sie den Wert einer Variablen abfragen oder einen neuen Wert in einer Variablen abspeichern wollen.

Nun, im Grunde ist es recht einfach. Nach der Definition repräsentiert der Variablenname grundsätzlich den Wert der Variablen. Nur wenn der Variablenname auf der linken Seite einer Zuweisung auftaucht, steht er rein für den Speicherplatz, in dem der Ergebniswert des rechten Teils der Zuweisung abzulegen ist.

In einer einzigen Zuweisung kann ein Variablenname also sowohl den Wert der Variablen als auch den Speicherplatz der Variablen repräsentieren:

```
summe = summe + 15;
```

Aus diesem Grunde spricht man auch vom L-Wert (Linkswert = Speicher der Variablen, in dem ein neuer Wert abgelegt werden kann) und R-Wert (Rechtswert, aktueller Wert der Variablen).

Implizite Variablendefinition durch Zuweisung

Variablen müssen in JavaScript nicht explizit mit `var` definiert werden. Jede Zuweisung kann zu einer Variablendefinition werden, wenn Sie auf der linken Seite der Zuweisung einen bisher noch nicht benutzten Variablennamen verwenden.

Statt

```
<script type="text/javascript">
  var summe = null;
  summe = 3 * 15;
  document.write("<p>");
  document.write(summe);
  document.write("</p>");
</script>
```

könnte man also auch schreiben:

```
<script type="text/javascript">
  summe = 3 * 15;
  document.write("<p>");
  document.write(summe);
  document.write("</p>");
</script>
```

Hey, dann können wir uns ja den ganzen formellen Kram mit der Variablendefinition sparen?! Tja, so war es wohl gedacht. Vermutlich waren die Erfinder von JavaScript der Auffassung, dass man es Webdesignern nicht zumuten kann, ihre Variablen vor dem Gebrauch zu definieren. Nur leider handelt man sich damit viel Ärger ein, denn der Interpreter kann nicht mehr zwischen einem Tippfehler und einer neu zu definierenden Variablen unterscheiden. Schauen wir uns als Beispiel eine Variante des Skriptes aus Listing 9.2 an:

```
<script type="text/javascript">

  var dm_wert    = null;
  var euro_wert = null;

  dm_Wert = prompt("Geben Sie einen Wert in DM an: ", "");
  euro_wert = dm_wert / 1.95583;
  document.write("<p>");
  document.write(dm_wert);
  document.write(" DM entsprechen ");
  document.write(euro_wert);
  document.write(" Euro.</p>");
</script>
```

Wenn dieses Skript ausgeführt wird, lautet die HTML-Ausgabe stets:

```
null DM entsprechen 0 Euro.
```

Was ist hier schief gelaufen?

Der Fehler liegt in der Zuweisung der Benutzereingabe an die vermeintliche Variable `dm_wert`. In der Zuweisung haben wir nämlich aus Versehen das `w` groß geschrieben:

```
dm_Wert = prompt("...
```

Da JavaScript streng zwischen Groß- und Kleinschreibung unterscheidet, bezieht der Interpreter den Bezeichner `dm_Wert` nicht auf die zuvor definierte Variable `dm_wert`. Da der neue Bezeichner `dm_Wert` auf der linken Seite einer Zuweisung auftaucht, erzeugt der Interpreter einfach eine neue Variable für den Bezeichner und weist dieser den Ergebniswert der `prompt`-Methode zu.

Da wir aber im weiteren Verlauf des Codes wieder auf die eigentliche Variable `dm_wert` zugreifen, wird der vom Webbesucher eingegebene Wert gar nicht beachtet.

In obigem Beispiel wird der Fehler beim Testen des Skriptes im Browser schnell offensichtlich. Manchmal sind solche Fehler aber nicht so einfach zu erkennen und führen unbemerkt zu falschen Ergebnissen. Man könnte solche Fehler vermeiden, wenn der Interpreter nur Variablen akzeptieren würde, die zuvor explizit deklariert worden sind. Vielleicht wird dies irgendwann in der Zukunft sogar der Fall sein. Bis dahin heißt es aufpassen und gewöhnen Sie sich schon einmal einen sauberen Programmierstil an, indem Sie alle Variablen vorab deklarieren.

Datentypen von Variablen

JavaScript unterscheidet zwischen vier Arten von Daten plus zwei Sondertypen.

Datentyp	Beschreibung

Boolean	Datentyp für Wahrheitswerte. Es gibt nur zwei erlaubte Werte, für die die Schlüsselwörter true (wahr) und false (falsch) definiert sind.
Zahl	Datentyp für Ganz- und Bruchzahlen.
String	Datentyp für Zeichenfolgen (Text)
Objekt	Strukturierter Datentyp für komplexere Daten, siehe 9.5.

Tabelle 9.3: JavaScript-Datentypen

Die Sondertypen umfassen jeweils nur einen Wert.

Wert/Datentyp	Beschreibung
undefined	Variablen, die definiert sind, denen aber noch kein Wert zugewiesen wurde, haben automatisch den Wert undefined.
null	Null-Wert. Programmierer können ihren Variablen diesen Wert zuweisen, wenn Sie anzeigen wollen, dass diese Variable noch keinen vernünftigen Wert enthält. Beispielsweise liefern die JavaScript-Methode prompt den null-Wert zurück, wenn der Webbesucher den Eingabedialog durch Drücken der Abbrechen-Taste schließt. Dank des null-Wertes kann der Programmierer also unterscheiden, ob der Webbesucher den Zahlenwert 0 eingegeben hat, oder ob er den Dialog abgebrochen hat.

Tabelle 9.4: Sondertypen

Da JavaScript eine nur »schwach typisierte« Programmiersprache ist, können Sie einer Variablen Werte jedes dieser Datentypen zuweisen, ja Sie können ein und derselben Variablen sogar nacheinander Werte verschiedener Datentypen zuweisen.

```
var meineVar = null;
meineVar = -123;
meineVar = "Hallo";
```

Leser, die von der C/C++- oder Java-Programmierung herkommen, wird hierbei das kalte Grausen erfassen, denn solcher Typenmischmasch ist in diesen Sprachen (aus guten Gründen) nicht erlaubt. Und auch wir als JavaScript-Programmierer sollten uns darum bemühen, unsere Variablen möglichst sauber nach Datentypen zu trennen. Das heißt, wenn wir eine Variable alter definiert haben, in der wir das vom Webbesucher eingegebene Alter abspeichern (also eine Zahl), sollten wir nicht weiter unten im Skript darauf verfallen, in der Variablen einen auszugebenden Text zwischenzuspeichern (selbst dann nicht, wenn die Altersangabe nicht mehr benötigt wird).



Jeder Variable kommt im Skript eine bestimmte Bedeutung zu. Vermeiden Sie es, Bedeutung oder Datentyp einer Variable im Verlaufe eines Skriptes zu ändern. Definieren Sie lieber eine weitere Variable.

Typenumwandlung

Manchmal ist es aber erforderlich, den Typ einer Variablen (oder eines Wertes) umzuwandeln. Betrachten wir dazu noch einmal den Code aus unserem Euro-Skript:

```
var dm_wert    = null;
var euro_wert  = null;

dm_wert = prompt("Geben Sie einen Wert in DM an: ", "");
euro_wert = dm_wert / 1.95583;
...
document.write(euro_wert);
...
```

Zu Beginn des Codes werden zwei Variablen `dm_wert` und `euro_wert` definiert und mit `null` initialisiert. Die Variablen gehören danach dem Sondertyp `null` an.

Dann wird vom Besucher der Webseite ein Zahlenwert abgefragt und in der Variablen `dm_wert` gespeichert. Benutzereingaben sind immer Strings, und folglich liefert die `prompt`-Methode auch einen String zurück. Wenn der Webbesucher beispielsweise 123 eingibt, liefert `prompt` nicht den Zahlenwert 123, sondern den String "123" zurück! Dieser String wird nun in `dm_wert` gespeichert. Der Typ der Variablen `dm_wert` ändert sich also vom Null-Typ in den String-Typ. Da der Null-Typ ein Sondertyp ist, ist dies nicht weiter aufregend. Richtig gefährlich wird es aber in der nächsten Zeile.

```
euro_wert = dm_wert / 1.95583;
```

Hier soll der Wert von `dm_wert` durch 1.95583 dividiert werden. `dm_wert` enthält aber einen String. Wie bitte kann man einen String (also einen Text) durch eine Zahl teilen? Hier schreitet der Interpreter ein. Er erkennt, dass eine Division durchzuführen ist, und da er weiß, dass man nur Zahlen dividieren kann, versucht er den String in eine Zahl umzuwandeln. Enthält der String tatsächlich eine brauchbare Zahlenangabe, so ist dies kein Problem. Und so wird aus dem String "123" die Zahl 123.



Diese Umwandlung gilt nur für den Wert im Ausdruck, das heißt, im Ausdruck steht `dm_wert` jetzt für die Zahl 123, in der Variablen `dm_wert` ist aber weiter der String "123" gespeichert.

Das Ergebnis der Division wird in der Variablen `euro_wert` abgespeichert, die danach dem Zahlen-Typ angehört.

Zum Schluss wird die Variable `euro_wert` ausgegeben. Ausgaben sind aber immer Strings, und so wandelt der Interpreter für die Ausgabe mit `write` den Wert von `euro_wert` in einen String um.

Binärcodierung von Daten

Sie sehen: die Realität ist wieder einmal viel komplizierter als es auf den ersten Blick erscheint. Der Grund hierfür ist, dass im Computer alle Daten als Bitfolgen gespeichert werden müssen (Bitfolgen sind Folgen von 0 und 1, was im Computer meist elektrische Spannung oder keine elektrische Spannung bedeutet).

Dabei verwendet der Computer für die einzelnen Datentypen unterschiedliche Codierungsverfahren. Aus diesem Grunde sieht die Bitfolge, die einen Zahlenwert 123 repräsentiert, ganz anders aus als die Bitfolge, die den String »123« repräsentiert. Während wir Menschen einen String »123« ohne Probleme auch als Zahl 123 interpretieren können¹, muss der JavaScript-Interpreter dazu erst die Bitfolgen ineinander umwandeln.

Glücklicherweise haben wir mit diesen Interna bei der Programmierung wenig zu tun, da der Interpreter die Daten bei Bedarf automatisch konvertiert (siehe oben).

Was aber, wenn der Interpreter nicht so konvertiert, wie wir es uns vorstellen? Nehmen wir zum Beispiel die Addition mit dem Plus-Operator. Im Gegensatz zur Division, die nur für Zahlen erlaubt ist, gilt die Addition für Zahlen und Strings². Was also passiert, wenn wir eine Zahl und einen String addieren?

```
var zahl1 = 111;
var zahl2 = null;
var summe = null;

zahl2 = prompt("Geben Sie eine Zahl ein: ", "");
summe = zahl1 + zahl2;
document.write(summe);
```

Beachten Sie, dass die Variable `zahl2` einen String-Wert enthält. (Nur weil wir die Variable hoffnungsfroh `zahl2` getauft haben, wird der Interpreter nicht dafür sorgen, dass diese Variable nur Zahlenwerte aufnimmt. Entscheidend ist hier, dass `prompt` einen String zurückliefert, der in `zahl2` gespeichert wird. Für die weitere Besprechung des Beispiels nehmen wir einfach an, der Webbesucher hat 222 eingegeben und in `zahl2` steht jetzt der String "222".

Bei der Addition trifft der Interpreter nun auf eine Zahl (`zahl1`) und einen String (`zahl2`). Was wird er tun? Wenn Sie das

Beispiel nachprogrammieren, werden Sie feststellen, dass der Interpreter zahl1 in einen String umwandelt, denn die Ausgabe dieses Codes lautet:

111222

Das ist sicherlich nicht das, was wir mit unserem Skript erreichen wollten. Wie kann man den Interpreter also dazu bringen, zahl2 in eine Zahl zu konvertieren?

Wenn der Interpreter die Werte von Variablen nicht so konvertiert, wie Sie es möchten, müssen Sie selbst für die richtige Konvertierung sorgen. Für die Konvertierung eines Strings in eine Zahl stellt uns JavaScript zwei window-Methoden zur Verfügung.

Funktion	Beschreibung
parseInt(string, basis)	Überprüft, ob der erste Teil eines Strings einen Integer enthält. Im zweiten Parameter wird die Basis des Zahlensystems angegeben. Wird keine Basis angegeben, liefert die Funktion eine Dezimalzahl zurück. a=parseInt("4 Kinder") liefert a = 4
parseFloat(string)	Entspricht in etwa parseInt(), sucht jedoch nach Fließkommazahlen. a="1.7 Kinder im Durchschnitt" liefert a = 1.7

Tabelle 9.5: Konvertierungsfunktionen

Damit die Addition wie gewünscht ausgeführt wird, müssen wir also schreiben:

```
summe = zahl1 + parseFloat(zahl2);
```

Gültigkeitsbereiche

Variablen kann man an beliebiger Stelle im Skriptcode definieren. Trotzdem ist es nicht ganz unerheblich, wo man eine Variable definiert, denn der Ort der Definition legt auch den Gültigkeitsbereich der Variable (im Englischen »scope«) fest.

Eine Variable kann entweder lokal, d.h. innerhalb einer Funktion, oder global, d.h. außerhalb einer Funktion deklariert werden. Lokale Variablen sind nur innerhalb der Funktion gültig, in der Sie definiert wurden. Globale Variablen sind ab dem Ort Ihrer Deklaration im aktuellen und allen nachfolgenden <script>-Tags gültig.

Im ersten Moment könnte man meinen, globale Variablen sind ein praktisches Mittel, um vorab am Anfang der Webseite einen Schwung passender globaler Variablen zu definieren, die man dann weiter unten in dem einen oder anderen <script>-Element oder der einen oder anderen JavaScript-Funktion verwendet. Doch dieser Missbrauch globaler Variablen führt zu nur schwer verständlichem Code und öffnet Bugs (liebevolle Bezeichnung für Programmierfehler) Tür und Tor. Versuchen Sie stattdessen Ihren Code so modular wie möglich zu halten, indem Sie für jede Funktion und jedes <script>-Element eigene Variablen definieren. Nur wenn Sie von mehreren <script>-Elementen oder JavaScript-Funktionen auf ein und dieselbe Variable zugreifen wollen, sollten Sie diese als globale Variable zugänglich machen.

Verdeckung und Überschreibung

Wenn Sie in einer Funktion eine lokale Funktion definieren, die den gleichen Namen wie eine zuvor definierte globale Variable trägt, verdeckt die lokale Variable in der Funktion die globale Variable. Das bedeutet, dass sich der Bezeichner innerhalb der Funktion immer auf die lokale Funktion bezieht, während außerhalb weiter die globale Variable verfügbar ist.

Listing 9.3: Auszug aus scope.html

```
<body>
<h1>Verdeckung</h1>
<script type="text/javascript">
  var meineVariable = 3;           // globale Variable
  document.write("<p>Wert von meinVariable in 1. Skriptelement: ");
  document.write(meineVariable);  // Ausgabe: 3
```

```

document.write("</p>");
</script>
<script type="text/javascript">
function funk()
{
var meineVariable = 5;           // lokale Variable
                                // verdeckt globale Variable
document.write("<p>Wert von meinVariable in Funktion: ");
document.write(meineVariable);  // Ausgabe: 5
document.write("</p>");
}
funk();
document.write("<p>Wert von meinVariable in 2. Skriptelement: ");
document.write(meineVariable);  // 3 (alte globale Variable)
document.write("</p>");
</script>
</body>

```

Wenn Sie dagegen in einem `<script>`-Element eine globale Variable definieren, die den gleichen Namen wie eine bereits zuvor definierte globale Variable trägt, überschreibt die später definierte globale Variable die zuvor definierte globale Variable. Der globale Bezeichner bezieht sich also auf die zuletzt definierte globale Variable.

9.2 Operatoren und Ausdrücke

Wie man Daten in Variablen zwischenspeichert, wissen wir jetzt. Der nächste Schritt ist, mit den Daten zu arbeiten, sie auszuwerten, sie zu verändern. Dies geschieht mit Hilfe von Operatoren. Einige Operatoren haben Sie bereits kennen gelernt, etwa den Zuweisungsoperator `=` oder den Additionsoperator `+`. Aber es gibt noch eine ganze Reihe weiterer Operatoren.

Arithmetische Operatoren

Operatoren	Funktion	Beispiel
+	Addition	3 + 4
	Verknüpfung von Strings	"Hallo" + "Joe"
-	Subtraktion	3 - 4
	Negation eines Wertes	-12
*	Multiplikation	3 * 4
/	Division	3 / 4
%	Modulo	4 / 3
++	Inkrement	++var
		var++
--	Dekrement	--var
		var--

Tabelle 9.6: Arithmetische Operatoren

Die arithmetischen Operatoren dienen vor allem der Bearbeitung von Zahlen und werden so eingesetzt, wie man es aus der algebraischen Mathematik kennt (einzige Ausnahme ist der `+`-Operator, mit dem man Strings aneinander hängen kann). Das folgende Skript fragt zum Beispiel vom Besucher der Webseite zwei Zahlenwerte ab und multipliziert diese mit Hilfe des `*`-Operators.



Da der *-Operator im Gegensatz zum +-Operator nicht für Strings definiert ist, können wir uns die explizite Typumwandlung der Eingaben sparen (siehe vorangehender Abschnitt zu Typumwandlung).

Listing 9.4: multiplizieren.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Operatoren</title>
</head>
<body>
<h1>Multiplikation</h1>
<script type="text/javascript">

  var zahl1 = prompt("Geben Sie eine Zahl ein: ","");
  var zahl2 = prompt("Geben Sie eine Zahl ein: ","");
  var produkt = zahl1 * zahl2;
  document.write("<p>");
  document.write(zahl1);
  document.write(" * ");
  document.write(zahl2);
  document.write(" ist gleich ");
  document.write(produkt);
  document.write("</p>");
</script>
</body>
</html>
```

Womöglich nicht ganz so bekannt wie die Multiplikation oder Addition sind Modulo, Inkrement und Dekrement.

Modulo

Der Modulo-Operator teilt seinen ersten Operanden ganzzahlig durch den zweiten Operanden und gibt den dabei verbleibenden Rest zurück.

Wenn Sie beispielsweise schreiben:

```
var zahl1 = 8;
var zahl2 = 3;
var produkt = zahl1 % zahl2;
```

wird in der Variablen produkt der Wert 2 abgespeichert. Der Interpreter teilt 8 durch 3 und stellt fest, dass bei einer Ganzzahldivision ein Rest von 2 bleibt. Diesen Wert liefert er als Ergebnis der Modulo-Operation zurück.



Der Modulo-Operator kann in JavaScript auch zusammen mit Bruchzahlen verwendet werden.

Inkrement und Dekrement

Eine bei der Programmierung häufig benötigte Operation ist das Erhöhen oder Vermindern des Werts einer Variablen um 1.³ Man bezeichnet dies als Inkrement (Erhöhung um 1) oder Dekrement (Verminderung um 1).

Damit man zur Inkrementierung einer Variablen nicht immer

```
var = var + 1;
```

schreiben muss, gibt es in JavaScript den Inkrementoperator:

```
++var;
```

Entsprechend schreibt man zur Dekrementierung einer Variablen:

```
--var;
```

Inkrement- und Dekrementoperator kann man der Variablen voranstellen (Präfixnotation: ++var) oder nachstellen (Postfixnotation: var++).

Wenn man die Operatoren alleine in einer Anweisung benutzt

```
++var;  
var++;
```

spielt es keine große Rolle, welche Version man verwendet.



Geschwindigkeitsfanatiker wird es interessieren, dass die Präfixnotation schneller ist, da der Interpreter für ihre Implementierung weniger Maschinenbefehle benötigt.

Wenn Sie die Operatoren aber in einem Ausdruck - beispielsweise auf der rechten Seite einer Zuweisung - verwenden, stellt sich die Frage, welcher Wert des Operanden für die Berechnung des Ausdrucks verwendet wird. Schauen Sie sich dazu die folgende Anweisung an:

```
var zahl1 = 12;  
zahl2 = 4 * ++zahl1;
```

Welchen Wert hat zahl2 nach Ausführung dieses Codes? Um diese Frage zu beantworten, muss man wissen, welchen Wert ++zahl1 in dem Ausdruck repräsentiert: den Wert, den zahl1 vor der Inkrementierung hat, oder den Wert, den die Variable nach der Inkrementierung hat?

- Die Präfixnotation erhöht den Wert der Variablen und gibt danach den neuen Wert zurück.
- Die Postfixnotation erhöht den Wert der Variablen, liefert aber den alten Wert zurück.

```
zahl1 = 12;  
zahl2 = 4 * ++zahl1;    // zahl2 gleich 4*13 = 52  
zahl1 = 12;  
zahl2 = 4 * zahl1++;    // zahl2 gleich 4*12 = 48
```

Zuweisungsoperatoren

Mit dem Zuweisungsoperator kann man in einer Variablen den Ergebniswert eines Ausdrucks abspeichern:

```
zahl1 = 3;  
zahl1 = 3 + 4;  
zahl2 = zahl1 * (3 + zahl1);  
zahl2 = zahl2 + 4;
```

Wir haben in unseren Skripten bereits fleißig von ihm Gebrauch gemacht.

Was wir Ihnen bisher verschwiegen haben, ist, dass es Kurzfassungen für Operationen gibt, in denen der neue Wert der

Variable auf ihrem alten Wert basiert. Solche Operationen haben wir bisher immer so formuliert, dass wir die Variable auf der linken und rechten Seite des Zuweisungsoperators verwendet haben:

```
zahl2 = zahl2 + 4; // Erhöht den Wert von zahl2 um 4
zahl2 = zahl2 * 4; // Multipliziert den Wert von zahl2 mit 4
```

Kürzer kann man diese Operationen mit Hilfe eines zusammengesetzten Zuweisungsoperators schreiben:

```
zahl2 += 4;
zahl2 *= 4;
```

Kombinierte Zuweisungsoperatoren gibt es für die arithmetischen Operationen:

`+=, -=, *=, /=, %=`

und die Bitoperatoren (siehe Abschnitt 9.2.4):

`|=, ^=, &=, <<=, >>=` und `>>>=`



Die kombinierten Zuweisungsoperationen sparen nicht nur Tipparbeit, meist werden sie auch schneller ausgeführt.

Vergleiche und logische Operatoren

Die letzte wirklich wichtige Gruppe von Operatoren sind die Vergleichsoperatoren. Das Vergleichen von Werten spielt in der Programmierung eine ganz besondere Rolle, denn auf der Grundlage solcher Vergleiche kann man den Ablauf eines Programms steuern. Wie dies geht werden Sie in Abschnitt 9.3 erfahren. Die zugehörigen Vergleichsoperatoren sollen Sie jetzt schon kennen lernen.

Vergleichsoperatoren

Operatoren	Funktion	Beispiel
<code>==</code>	Gleichheit Liefert true, wenn beide Werte gleich sind.	<code>var1 == var2</code>
<code>!=</code>	Ungleichheit Liefert true, wenn beide Werte ungleich sind.	<code>var1 != var2</code>
<code><</code>	Kleiner Liefert true, wenn der erste Wert kleiner ist als der zweite	<code>var1 < var2</code>
<code><=</code>	Kleiner gleich Liefert true, wenn der erste Wert kleiner oder gleich dem zweiten ist	<code>var1 <= var2</code>
<code>></code>	Größer Liefert true, wenn der erste Wert größer ist als der zweite	<code>var1 > var2</code>
<code>>=</code>	Größer gleich Liefert true, wenn der erste Wert größer oder gleich dem zweiten ist	<code>var1 >= var2</code>

Tabelle 9.7: Vergleichsoperatoren

Allen Vergleichsoperatoren gemein ist, dass Sie sowohl für Boolesche Werte, Zahlen oder auch Strings verwendet werden können und als Ergebnis einen Booleschen Wert zurückliefern (true oder false), je nachdem, ob die Behauptung des Vergleichs wahr oder falsch ist.

Wie werden Strings verglichen?

Strings werden lexikographisch verglichen, das heißt, der Interpreter geht die Strings parallel Zeichen für Zeichen von vorne nach hinten durch, bis sich die Strings in einem Zeichen unterscheiden. Dann prüft er, welches Zeichen im Alphabet früher kommt. Der String mit diesem Zeichen ist dann kleiner als der andere String.

Genau genommen prüft der Interpreter nicht nach der Reihenfolge im Alphabet, sondern nach der Reihenfolge der Zeichen in der Unicode-Tabelle.

Kompliziert kann es werden, wenn eine Zahl mit einem String verglichen wird. In solchen Fällen kann man einen arithmetischen Zahlenvergleich beziehungsweise einen lexikographischen Stringvergleich erzwingen, indem man die zu vergleichenden Werte mit Hilfe eines Tricks entsprechend umwandelt:

```

(" " + var1) == (" "+ var2)           // erzwingt einen Stringvergleich
(var1 - 0) == (var2 - 0)              // erzwingt einen Zahlenvergleich

```



JavaScript kennt laut ECMA noch zwei weitere Vergleichsoperatoren: `===` und `!==`. `===` liefert false, wenn die beiden verglichenen Werte unterschiedlichen Datentypen angehören, `!==` liefert in so einem Fall entsprechend true.

Logische Operatoren

Mit Hilfe der folgenden Operatoren kann man die Ergebnisse von Vergleichen kombinieren oder umwandeln.

Operatoren	Funktion	Beispiel
!	Logisches NICHT Wandelt den Ergebniswert eines Vergleichs um. Liefert der Vergleich also beispielsweise true, macht der !-Operator daraus false.	!(var1 == var2)
&&	Logisches UND Liefert true, wenn beide kombinierten Vergleiche true liefern.	(var1 < var2) && (var3 > var2)
	Logisches ODER Liefert true, wenn mindestens einer der beiden kombinierten Vergleiche true liefert.	(var1 < var2) (var3 > var2)

Tabelle 9.8: Vergleichsoperatoren

Bitweise Operatoren

Mit Hilfe der Operatoren `&`, `|`, `^`, `<<`, `>>` und `>>>` kann man Werte auf Bitebene manipulieren. Diese Operatoren sollten erfahrenen Programmierern vorbehalten bleiben, weswegen wir auf diese Operatoren, die im Übrigen ganz wie in anderen Programmiersprachen verwendet werden, nicht näher eingehen werden.

Ausdrücke und Anweisungen



Eine Kombination aus Werten, Variablen und Operatoren, die vom Interpreter zu einem Ergebniswert berechnet werden kann, bezeichnen Programmierer als Ausdruck.

Gültige Ausdrücke sind beispielsweise:

```
var1
var1 * 3
3 < var2
1 + 2 + 3 + var1 * var2
```

Ausdrücke findet man beispielsweise auf der rechten Seite von Zuweisungen, in Bedingungen von Verzweigungen und Schleifen (siehe Abschnitt 9.3) oder als Argumente in Funktionsaufrufen (siehe Abschnitt 9.4.2).

Da in einem Ausdruck mehrere Operatoren vorkommen können, ist es wichtig zu wissen, in welcher Reihenfolge die Operatoren abgearbeitet werden. Die folgende Aufzählung beginnt mit den vorrangig ausgeführten Operationen und geht bis zu der stets zuletzt ausgeführten Zuweisung.

- Negationen
- Multiplikationen, Divisionen, Modulo-Operationen
- Additionen und Subtraktionen
- Vergleiche
- Logische Operatoren
- Zuweisung

Gefällt Ihnen diese Reihenfolge nicht, können Sie sie durch das Setzen von Klammern verändern.

Machen wir einen kleinen Test. Wie sieht zum Beispiel das Ergebnis folgender Berechnung aus?

```
var zahl1 = 3;
var zahl2 = 2;
var zahl3 = 7;
var ergebnis = zahl1 + zahl2 * zahl3;
```

Es lautet 17, weil die Multiplikation von `zahl2 * zahl3` vor der Addition ausgeführt wird. Wenn Sie möchten, dass zuerst die Addition ausgeführt und das Ergebnis dann mit `zahl3` multipliziert wird, müssen Sie schreiben:

```
var zahl1 = 3;
var zahl2 = 2;
var zahl3 = 7;
var ergebnis = (zahl1 + zahl2) * zahl3;
```

Schon lautet das Ergebnis 35.

Anweisungen

Ein Ausdruck ist letztendlich nicht mehr als Wert, den der Interpreter nach einer mehr oder weniger komplizierten Formel berechnet. Ein Ausdruck ist aber noch kein Befehl an den Computer. Befehle bezeichnet man in der Programmiersprache als Anweisungen und man erkennt sie daran, dass sie stets mit einem Semikolon ; abgeschlossen werden. Einige Anweisungen haben Sie bereits kennen gelernt,

beispielsweise die Variablendeklaration:

```
var meineZahl1, meineZahl2;
```

oder die Zuweisung:

```
meineZahl1 = 3 * meineZahl2;
```

Anweisungen können mittels geschweifter Klammern zu Blöcken zusammengefasst werden. Wir kennen dies bereits aus der Funktionsdefinition, wo alle Anweisungen, die zu der Funktion gehören, in geschweiften Klammern stehen:

```
function fkt_name()  
{  
  document.bgColor = "cyan";  
}
```

9.3 Steuerung des Programmablaufs

Anweisungen werden standardmäßig nacheinander von oben nach unten abgearbeitet und ausgeführt. Meist ist dies auch genau das, was wir wollen. Manchmal jedoch würde man sich etwas mehr Flexibilität wünschen, beispielsweise wenn man in Abhängigkeit von dem Wert einer Variablen verschiedene Anweisungen ausführen lassen möchte oder wenn man eine bestimmte Anweisung mehrere Male hintereinander ausführen lassen möchte.

Für solche Aufgaben stellt JavaScript verschiedene Kontrollstrukturen zur Verfügung, mit denen man den Programmablauf steuern kann.



Technisch gesehen zählen diese Kontrollstrukturen auch zu den Anweisungen, auch wenn sie meist nicht direkt mit einem Semikolon abgeschlossen werden.

Die if-Verzweigung

Die if-Verzweigung entscheidet, ob ein nachfolgender Anweisungsblock ausgeführt werden soll oder nicht. Dazu wird - meist mit Hilfe der Vergleichsoperatoren (siehe Abschnitt 9.2.3) - eine Bedingung formuliert, die vom Programm zur Laufzeit ausgewertet wird. Ist die Bedingung erfüllt (true), wird der auf die Bedingung folgende Block ausgeführt, ist die Bedingung falsch (false), wird der Block übersprungen und das Programm wird mit der nächsten Anweisung hinter der if-Verzweigung fortgeführt.

Die allgemeine Syntax der if-Bedingung sieht wie folgt aus:

```
if (Bedingung)  
{  
  Anweisung(en);  
}
```

Diese Konstruktion kann man wie einen deutschen Konditionalsatz lesen:

»Wenn die Bedingung erfüllt ist, dann (und nur dann) führe die Anweisungen aus«

Wenn Sie beispielsweise die Wurzeln von Zahlen berechnen, müssen Sie beachten, dass die Wurzel nur für positive Zahlen definiert ist.

Listing 9.5: if.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"  
  "http://www.w3.org/TR/html4/strict.dtd">  
<html>  
<head>  
  <title>if-Verzweigung</title>  
</head>  
<body>  
<h1>Wurzelberechnung</h1>  
<script type="text/javascript">
```

```

var eingabe;
var wurzel;
eingabe = prompt("Von welchem Wert wollen Sie die Wurzel berechnen: ",
                "");
if (eingabe >= 0)
{
    wurzel = Math.sqrt(eingabe);
    document.write("<p>Die Wurzel von ");
    document.write(eingabe);
    document.write(" ist ");
    document.write(wurzel);
    document.write("</p>");
}
</script>
</body>
</html>

```



Abbildung 9.2: Ausgabe des Codes aus if.html

Da die Verwendung der if-Verzweigung nicht schwer zu verstehen ist, möchte ich Ihre Aufmerksamkeit hier auf die Berechnung der Wurzel lenken. Viele mathematische Funktionen, die uns allen so geläufig sind, dass man verleitet ist, sie als »einfache« Operationen anzusehen (beispielsweise der Sinus oder die Wurzel), sind in Wirklichkeit nur schwierig und mit Hilfe komplizierter mathematischer Algorithmen zu berechnen. Da es keinem Programmierer zuzumuten ist, diese Algorithmen selbst zu implementieren, legt der ECMA-Standard fest, dass jeder JavaScript-Interpreter ein `Math`-Objekt zu Verfügung stellen soll, das die wichtigsten mathematischen Funktionen als Methoden implementiert. So heißt die Methode für die Berechnung der Wurzel beispielsweise `sqrt()` (steht für »square root«) und liefert als Ergebnis die Wurzel des übergebenen Arguments zurück.

```
wurzel = Math.sqrt(eingabe);
```



Soll die if-Bedingung nur eine einzige Anweisung kontrollieren, brauchen Sie diese nicht in geschweifte Klammern einzuschließen.



Schließen Sie die if-Bedingung nie, wirklich nie, mit einem Semikolon ab!!! Der Interpreter denkt dann, dass das Semikolon für eine leere Anweisung (eine Anweisung, die nichts macht) stehe und dass die if-Bedingung diese

leere Anweisung kontrolliere.

Die else-Alternative

Mit einer if-else-Konstruktion kann man in Abhängigkeit von einer Bedingung den einen oder den anderen Block ausführen lassen. Wir wollen uns dies in unserem Wurzel- Programm zunutze machen, indem wir den Anwender im Falle negativer Eingaben gezielt auf seinen Fehler hinweisen.

Listing 9.6: Auszug aus else.html

```
<script type="text/javascript">
  var eingabe;
  var wurzel;
  eingabe = prompt("Von welchem Wert wollen Sie die Wurzel berechnen: ",
    "");
  if (eingabe >= 0)
  {
    wurzel = Math.sqrt(eingabe);
    document.write("<p>Die Wurzel von ");
    document.write(eingabe);
    document.write(" ist ");
    document.write(wurzel);
    document.write("</p>");
  }
  else
  {
    document.write("<p>Sie haben die Zahl ");
    document.write(eingabe);
    document.write(" eingegeben.</p>");
    document.write("<p>Die Wurzel kann aber nur für positive Zahlen /
    berechnen werden.</p>");
  }
</script>
```



Abbildung 9.3: Ausgabe von else.html



Soll der else-Teil nur eine einzige Anweisung kontrollieren, brauchen Sie diese nicht in geschweifte Klammern einzuschließen.

Die switch-Verzweigung

Will man die Programmausführung in Abhängigkeit vom Wert einer Variablen mehrfach aufspalten, bietet sich dazu eine switch-Konstruktion an.

```
switch(variable)
{
case Wert1: Anweisung(en); break;
case Wert2: Anweisung(en); break;
case Wert3: Anweisung(en); break;
case Wert4: Anweisung(en); break;
default:    Anweisung(en);
}
```

Die switch-Anweisung vergleicht den Wert in der Variablen mit den Werten der case- Marken. Wird eine Übereinstimmung gefunden, springt die Programmausführung zur ersten Anweisung neben der case-Marke. Von dort ab, werden alle nachfolgenden Anweisungen in der switch-Verzweigung ausgeführt. Will man verhindern, dass der Code nachfolgender case-Marken ebenfalls ausgeführt wird, muss man die Anweisungsfolgen der case-Marken mit einer break-Anweisung abschließen. Diese sorgt dafür, dass die Programmausführung zum Ende der switch-Verzweigung springt.

Wird keine Übereinstimmung gefunden, springt die Programmausführung in den default- Block (sofern ein solcher aufgesetzt wurde).

Listing 9.7: switch.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>switch-Verzweigung</title>
</head>
<body>
<script type="text/javascript">

var alter;
alter = prompt("In welche Altersklasse fallen Sie? \n \
              1 (5-18), 2 (19-65) oder 3 (66-100)", "");
switch(parseInt(alter))
{
case 1:  document.bgColor = "pink";
        document.write("<h1>Hi, willkommen auf meiner
                        -Website</h1>");
        break;
case 2:  document.bgColor = "cyan";
        document.write("<h1>Hallo und willkommen auf meiner
                        -Website</h1>");
        break;
case 3:  document.bgColor = "white";
        document.write("<h1>Willkommen auf meiner Website</h1>");
        break;
default: document.bgColor = "white";
        document.write("<h1>Willkommen auf meiner Website</h1>");
        break;
}
</script>
</body>
</html>
```



Abbildung 9.4: Ausgabe nach Eingabe von 1

Mit Hilfe der switch-Anweisung kann die Programmausführung sehr gut in Abhängigkeit von den verschiedenen Werten einer Variablen aufgespalten werden. Sie unterliegt aber auch gewissen Beschränkungen:

- Man kann keine Teilbereiche abfragen. Formulierungen wie case 1-18: sind nicht möglich!
- Man kann keine komplexeren Vergleiche anstellen (hierfür nutzt man die if-else- Verzweigung, die man übrigens auch verschachteln kann).

An sich ist die Programmierung mit der switch-Verzweigung nicht sonderlich kompliziert, es gibt aber zwei Fallstricke, über die man als Anfänger leicht stolpern kann:

- Schließen Sie die Anweisungen zu einer case-Marke immer mit einer break- Anweisung ab, damit nicht die Anweisungen der nachfolgenden case-Marken ausgeführt werden.
- Achten Sie auf den Typ der switch-Variablen und wählen Sie entsprechend die case- Konstanten. Für Boolesche Variablen können Sie beispielsweise als case-Konstanten true und false verwenden (obwohl switch-Verzweigungen mit Booleschen Variablen eher ungewöhnlich sind und durch if-else-Verzweigungen ersetzt werden können), für numerische Variablen geben Sie Zahlenkonstanten und für String-Variablen Strings in Anführungszeichen an. Beachten Sie auch, dass Zahlen, die Sie über Formularfelder oder die prompt-Eingabeaufforderung anfordern, als Strings zurückgeliefert werden. Sie müssen die Eingaben daher entweder mit Strings ("1", "2", etc.) vergleichen oder die Eingabe in eine Zahl umwandeln (siehe Listing).



Passen Sie auf, von welchem Typ die Variable in der switch-Bedingung ist, und formulieren Sie entsprechend die case-Marken.

Die for-Schleife

Schleifen dienen dazu einen Anweisungsblock mehrfach hintereinander auszuführen. Eine der am häufigsten verwendeten Schleifen ist die for-Schleife.

```
for (Initialisierung; Bedingung; Veränderung)
{
    Anweisung(en);
}
```



Wie im Falle der *if*-Bedingung gilt: Kein Semikolon hinter dem Schleifenkopf!

Mit Hilfe einer solchen Schleife kann man beispielsweise bequem die ersten zehn Quadratzahlen berechnen und ausgeben.

Listing 9.8: for.html - Anweisungen mehrfach ausführen lassen

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>for-Schleife</title>
</head>
<body>
<h1>Die Quadratzahlen</h1>
<script type="text/javascript">

  var quadrat;
  var loop;
  for (loop=1; loop <= 10; ++loop)
  {
    quadrat = loop * loop;
    document.write("<p>Das Quadrat von ");
    document.write(loop);
    document.write(" ist ");
    document.write(quadrat);
    document.write("</p>");
  }
</script>
</body>
</html>
```



Abbildung 9.5: Ausgabe der JavaScript-Schleife

Die Schleife beginnt in der Zeile:

```
for (loop=1; loop <= 10; ++loop)
```

Eingeleitet wird die Schleife durch das Schlüsselwort *for*. In der nachfolgenden Klammer wird festgelegt, wie oft die Schleife durchlaufen wird. Dazu bedarf es einer Schleifenvariablen, die hier »loop« genannt wurde. Bei Eintritt in die Schleife muss der Schleifenvariablen ein Anfangswert zugewiesen werden. Dies geschieht in der Anweisung:

```
loop = 1;
```

Dahinter ist die Bedingung formuliert, die festlegt, wie lange die Schleife ausgeführt wird:

```
loop <= 10;
```

Solange die Bedingung erfüllt ist, das heißt, solange die Schleifenvariable loop kleiner oder gleich 10 ist, wird die Schleife ausgeführt.

Die dritte Anweisung wird nach jeder Abarbeitung des Schleifenkörpers ausgeführt. Sie zählt die Schleifenvariable hoch, so dass irgendwann die Schleifenbedingung nicht mehr erfüllt ist und die Schleife verlassen werden kann.

Der Schleifenkörper ist durch die geschweiften Klammern markiert.

Ausführung von Schleifen

Beim ersten Schleifendurchgang hat die Schleifenvariable loop noch den Anfangswert 1. Folglich bekommt quadrat den Wert $1 * 1 = 1$ zugewiesen. Das Ergebnis wird mit Hilfe der nachfolgenden write-Aufrufe ausgegeben.

Nach Abarbeitung der Anweisungen im Schleifenkörper wird die dritte Anweisung des Schleifenkopfes ausgeführt.

```
++loop
```



Bitte beachten Sie, dass die letzte Anweisung im Schleifenkopf nicht mit einem Semikolon abgeschlossen wird.

Danach springt das Programm zurück an den Schleifenanfang. Die Zuweisung

```
loop = 1;
```

wird nun nicht mehr ausgeführt, wohl aber die Überprüfung der Bedingung. Da nach jedem Schleifendurchgang die Variable loop um 1 erhöht wird, wird loop nach dem zehnten Durchgang den Wert 11 haben. Die Überprüfung der Bedingung ergibt, dass loop nun größer als 10 ist, und die Schleife wird verlassen.

Fallstricke

Wichtig ist, dass die Schleife auch wirklich wieder verlassen wird. Betrachten Sie folgende Schleife:

```
for(loop=1; loop <= 10; --loop)
{
    ...
}
```

Hier wird die Schleifenvariable loop nach jedem Schleifendurchgang dekrementiert, das heißt, um 1 vermindert. Der Wert von loop wird nie größer als 10 werden, und die Schleife wird nie verlassen!

Solche Endlosschleifen sind ein häufiges Beispiel für logische Fehler innerhalb eines syntaktisch korrekten Programms.

Die while-Schleife

Eine weitere wichtige Schleife ist die while-Schleife. Die while-Schleife wird solange wiederholt ausgeführt, wie die Bedingung neben dem Schlüsselwort while erfüllt ist.

```
Initialisierung;
while(Bedingung)
{
    Anweisung(en); // inklusive Veränderung
}
```

Ein mögliches Einsatzgebiet für die while-Schleife ist die Abfrage von Benutzereingaben, die einem bestimmten Wertebereich angehören müssen. Durch die while-Schleife kann man die Eingabe so lange wiederholen lassen, bis der Anwender einen korrekten Wert eingibt. In dem folgenden Programm, das eine Abwandlung des Wurzelprogramms aus Listing 9.5 ist, machen wir uns dies zunutze.

Listing 9.9: while.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>while-Schleife</title>
</head>
<body>
<h1>Wurzelberechnung</h1>
<script type="text/javascript">

  var eingabe = -1;
  var wurzel;
  while (eingabe < 0)
  {
    eingabe = prompt("Von welchem Wert wollen Sie die
                    -Wurzel berechnen: ", "");
  }
  wurzel = Math.sqrt(eingabe);
  document.write("<p>Die Wurzel von ");
  document.write(eingabe);
  document.write(" ist ");
  document.write(wurzel);
  document.write("</p>");
</script>
</body>
</html>
```

Die while-Schleife ist flexibler als die for-Schleife, aber auch gefährlicher, denn im Vergleich zur for-Schleife, wo alle Anweisungen, die die Schleifenausführung steuern, im Schleifenkopf zusammengefasst sind (Initialisierung, Prüfung und Veränderung der Schleifenvariable), sind diese Anweisungen bei der while-Schleife mehr oder weniger verstreut und der Interpretierer merkt auch nicht, wenn der Programmierer den einen oder anderen Teil vergessen hat. Sie sollten bei Verwendung einer while-Schleife sorgfältig prüfen, ob diese ordnungsgemäß betreten und irgendwann auch wieder verlassen wird. Wir wollen dies einmal an obigem Listing durch exerzieren.

Nehmen wir an, das Skript wird gerade gestartet und der Interpretierer trifft auf die while- Schleife.

```
while (eingabe < 0)
```

Beim ersten Mal soll der Interpretierer auf jeden Fall die Schleife ausführen, damit der Besucher zur Eingabe einer Zahl aufgefordert wird. Der Interpretierer führt die Schleife aber nur dann aus, wenn die Variable eingabe einen Wert kleiner Null enthält. Ist dies der Fall? Glücklicherweise ja, denn genau aus diesem Grunde haben wir die Variable eingabe weiter oben mit dem Wert -1 initialisiert.

```
var eingabe = -1;
```

Der Interpretierer führt jetzt den Code in der Schleife aus, das heißt, der Besucher wird aufgefordert eine Zahl einzugeben und diese wird in der Variablen eingabe abgespeichert. Danach ist der Schleifenkörper das erste Mal komplett abgearbeitet und die Programmausführung springt zurück zur Schleifenbedingung. Angenommen der Besucher hat einen negativen Wert eingegeben. Dann ist die Schleifenbedingung immer noch erfüllt und der Schleifenkörper wird erneut ausgeführt. So wird der Besucher wieder und wieder aufgefordert einen Wert einzugeben, bis er einen positiven Wert eintippt. Dann ist die Schleifenbedingung nicht mehr erfüllt, die Schleife wird nicht mehr ausgeführt und die Programmausführung wird mit der Berechnung der Wurzel fortgesetzt.

Abbruchbedingungen für Schleifen

Die Ausführung einer Schleife kann nicht nur durch das fein abgestimmte Spiel von Schleifenvariable und Schleifenbedingung, sondern auch mit Hilfe spezieller Anweisungen beeinflusst werden.

Anweisung	Beschreibung
continue	Bricht den aktuellen Schleifendurchgang ab und beginnt direkt mit dem nächsten Schleifendurchgang.
break	Bricht eine Schleife mitten in der Ausführung ab und fährt mit der nächsten Anweisung hinter der Schleife fort.

Tabelle 9.9: Abbruchanweisungen für Schleifen

Weitere Schleifen

Neben der for- und der while-Schleife gibt es noch zwei weitere Schleifenformen.

schleife	Beschreibung
do ... while	<pre>do { Anweisung(en) } while (Bedingung)</pre> <p>Diese Schleife unterscheidet sich von der while-Schleife dadurch, dass ihr Schleifenkörper unabhängig von der Schleifenbedingung mindestens einmal ausgeführt wird (wird von älteren Browsern nicht unterstützt)</p>
for ... in	Dies ist eine Sonderform der for-Schleife, mit der man bequem alle Elemente in einem Array durchlaufen kann (siehe Abschnitt 9.6.1)

Tabelle 9.10: Weitere JavaScript-Schleifen

9.4 Funktionen

Das Konzept der Funktionen ist nicht mehr ganz neu für uns. Von verschiedenen vordefinierten Funktionen wie write(), parseInt() oder sqrt() haben wir bereits fleißig Gebrauch gemacht⁴ und im vorangehenden Kapitel haben wir sogar bereits eigene Funktionen für die Ereignisbehandlung erstellt. In diesem Abschnitt wollen wir unser praktisches Wissen auf eine fundierte theoretische Basis stellen.

Die Funktionsdefinition und -aufruf

Eine Funktion zu definieren, bedeutet letzten Endes nichts anderes, als einen Anweisungsblock, der eine bestimmte Aufgabe erfüllt (beispielsweise einen Text ausgibt, die Wurzel einer Zahl berechnet oder den Antwortcode zu einem HTML-Ereignis enthält), mit einem Namen, dem Funktionsnamen, zu verbinden.

Der Vorteil dieses Verfahrens ist, dass man die Funktion nur einmalig irgendwo zu definieren braucht (bevorzugt am Anfang des HTML-Dokuments oder der JavaScript- Datei) und sie später durch Aufruf über ihren Namen ausführen lassen kann. Wir werden dies gleich an einem Beispiel verdeutlichen., zuvor aber noch die formelle Syntax der Funktionsdefinition.

Funktionsdefinition

```
function demo()
{
  Anweisung(en);
}
```

Funktionsdefinitionen werden mit dem Schlüsselwort function eingeleitet. Darauf folgt der Funktionsname, der mit runden Klammern abgeschlossen wird. (Weiter unten werden wir sehen, wie man in den runden Klammern eine Liste von Parametern definiert, über die die Funktion Werte entgegennehmen kann.) An diesen Funktionskopf schließen sich in geschweiften Klammern die auszuführenden Befehle an.

Funktionsaufruf

Funktionen werden einfach über ihren Namen aufgerufen.

```
demo();
```

Beispiel

Das folgende Beispiel enthält eine Webseite mit drei Textabsätzen. Jedes Mal, wenn der Besucher der Webseite mit der Maus über einen dieser Textabsätze fährt, wollen wir die Hintergrundfarbe der Webseite ändern. Der JavaScript-Code hierfür sieht wie folgt aus:

```
var f = parseInt( 1 + Math.random() * 5 );
switch(f)
{
  case 1: document.backgroundColor = "red"; break;
  case 2: document.backgroundColor = "green"; break;
  case 3: document.backgroundColor = "yellow"; break;
  case 4: document.backgroundColor = "cyan"; break;
  case 5: document.backgroundColor = "blue"; break;
}
```

Zuerst wird eine Zufallszahl zwischen 1 und 5 erzeugt. Zufallszahlen zu erzeugen, ist recht schwer. Wir greifen daher auf die Methode `Math.random` zurück, die uns eine zufällige Zahl zwischen 0 und 1 zurückliefert.

Zufallszahlen

Echte Zufallszahlen gibt es in der Programmierung nicht. Man kann aber Zahlenfolgen generieren, die sich erst sehr spät wiederholen und kein Muster erkennen lassen. Wenn man nacheinander auf die Zahlen eine solchen Zahlenfolge zugreift, kommt dies dem Ziehen von Zufallszahlen ziemlich nahe. Und genau dies tut die Methode `Math.random`.

Die von `Math.random` zurückgelieferte Zahl multiplizieren wir mit 5 (jetzt haben wir eine Zufallszahl zwischen 0 und 5). Dann addieren wir noch 1 (jetzt haben wir eine Zufallszahl zwischen 1 und 6). Zum Schluss wandeln wir die Zahl in eine Ganzzahl um und weisen diese `f` zu. Die Variable `f` enthält jetzt einen der Wert 1, 2, 3, 4 oder 5.

Mit Hilfe einer `switch`-Verzweigung verbinden wir jeden dieser Werte mit einer Hintergrundfarbe.

Jetzt müssen wir diesen Code nur noch mit dem `onmouseover`-Ereignis der drei Textabsätze verbinden. Wir könnten den Code dazu direkt als Wert des `onmouseover`-Attributs eintippen, doch bedeutet dies unnötige Tipparbeit und macht den Code der Webseite wegen der Vermischung von HTML- und JavaScript-Code sehr unübersichtlich. Besser ist es, den Code in Form einer Funktion am Anfang der Webseite zu definieren. In den `onmouseover`-Attributen brauchen wir dann nur noch die Funktion aufzurufen.

Listing 9.10: funktionen1.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>

  <title>Funktionen</title>
  <meta http-equiv="Content-Script-Type" content="text/javascript" />
  <script type="text/javascript">
function farbe_setzen()
{
  var f = parseInt( 1 + Math.random() * 5 );
  switch(f)
  {
    case 1: document.backgroundColor = "red"; break;
    case 2: document.backgroundColor = "green"; break;
    case 3: document.backgroundColor = "yellow"; break;
    case 4: document.backgroundColor = "cyan"; break;
  }
}
```

```

        case 5: document.bgColor = "blue"; break;
    }
}
</script>
</head>
<body>
<p onmouseover="farbe_setzen()">Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text ... </p>
<p onmouseover="farbe_setzen()">Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text ... </p>
<p onmouseover="farbe_setzen()">Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text ... </p>
</body>
</html>

```



Im Navigator 4 funktioniert dieses Beispiel so nicht, da der Navigator (im Gegensatz zum Internet Explorer oder Netscape 6-Browser) keine Mausereignisse für Absätze unterstützt. Wenn Sie mit dem Navigator arbeiten, ändern Sie die <p>-Absätze in Hyperlinks der Form: Text....

Die Auslagerung des Codes in eine Funktion hat noch einen weiteren Vorteil. Stellen Sie sich vor, Sie wollten die Farbtabelle erweitern oder einfach andere Farben vorsehen. In obigem Listing müssen Sie dazu nur einmal den Code der Funktion überarbeiten. Hätten Sie den Code dreimal in den <p>-Tags der Absätze stehen, müssten Sie die gleichen Änderungen dreimal vornehmen.



Es gibt noch andere Tricks, wie man zufällige Farben erzeugt. In Übung 3 werden Sie einen möglichen Algorithmus kennen lernen.

Parameter an Funktionen übergeben

Unsere in obigem Abschnitt definierte Funktion `farbe_setzen()` ist eine extrem unkommunikative Funktion, denn sie erlaubt keinen Datenaustausch zwischen sich und dem aufrufenden Code. Viele Funktionen werden aber erst dadurch wirklich wertvoll, dass sie vom Aufrufer zu verarbeitende Daten übernehmen und unter Umständen sogar Ergebniswerte zurückliefern. Wir kennen dieses Verhalten bereits von den vordefinierten Methoden.

- `document.write()` ist beispielsweise eine Methode, die die zu verarbeitenden Daten (den auszugebenden Text) beim Aufruf übernimmt.

```
document.write("<p>Hallo JavaScript!</p>");
```

- `parseInt()` ist eine Methode, die die zu verarbeitenden Daten (beispielsweise den umzuwandelnden Text) beim Aufruf übernimmt und das Ergebnis zurückliefert.

```
var zahl = parseInt("123");
```

In diesem und dem folgenden Abschnitt werden Sie sehen, wie Sie selbst Funktionen schreiben können, die beim Aufruf Werte übernehmen und Ergebnisse zurückliefern.

Soll eine Funktion beim Aufruf einen Wert übernehmen, muss man in der Funktionsdefinition in den runden Klammern einen Variablennamen für diesen Wert angeben.

Parameter und Argumente

Die in den runden Klammern definierten Variablen nennt man Parameter der Funktion. Sie sind nur in der Funktion gültig. Die Werte, die beim Aufruf der Funktion an diese Parameter übergeben werden, bezeichnet man als Argumente. (Es gibt aber auch viele Programmierer und Buchautoren, die in beiden Fällen von Parametern sprechen.)

```
function farbe_setzen(farbe)
{
  document.bgColor = farbe;
}
```

Diese Funktion dient dazu, die Hintergrundfarbe der Webseite zu setzen. Die Farbe übernimmt sie als Argument. Dadurch, dass die Hintergrundfarbe beim Aufruf angegeben werden kann, ist die Funktion recht flexibel einsetzbar (abgesehen davon, dass man für so triviale Aufgaben wie das Setzen der Hintergrundfarbe keine Funktion benötigt).

Aufruf:

```
farbe_setzen("red");
```

oder

```
onereignis="farbe_setzen('red')"
```

Call by value

Als Argumente kann man Konstanten oder Variablen übergeben. Wenn Sie Variablen übergeben, stellt sich die Frage, wie die Variable an die Funktion übergeben wird. Eine Möglichkeit wäre, dass die Variable selbst an die Funktion weitergereicht wird. Parameter und Variable wären dann identisch und die Funktion könnte über den Parameter den Wert der Variablen ändern. Man bezeichnet dies als call by reference, und wir sagen Ihnen gleich, dass dieses Verfahren von JavaScript nicht verwendet wird. JavaScript verwendet stattdessen das call by value-Verfahren, bei dem der Parameter als Kopie der Variablen erstellt wird. Das heißt, beim Aufruf wird der Wert der Variablen in den Parameter kopiert. Wenn später in der Funktion dem Parameter ein neuer Wert zugewiesen wird, ändert sich nur der Wert des Parameter, nicht der Wert der Variablen, die im Aufruf angegeben wurde.

Mehrere Parameter

Eine Funktion kann auch mehrere Parameter übernehmen. Diese werden dann durch Kommata getrennt in den runden Klammern aufgelistet.

```
function demo(par1, par2, par3)
{
  Anweisung(en);
}
```

Beim Aufruf der Funktion werden den Parametern die passenden Argumente übergeben: ebenfalls durch Kommata getrennt und in der gleichen Reihenfolge.

Werte aus Funktionen zurückliefern

So einfach die Übergabe von Werten an die Funktion ist, so einfach ist auch das Zurückliefern eines Wertes aus der Funktion. Sie brauchen dem zurückzuliefernden Wert lediglich das Schlüsselwort return voranzustellen:

```
return ergebnis_wert;
```

Das folgende Listing ist eine überarbeitete Version des Listing 9.8.

Listing 9.11: funktionen2.html - Werte übernehmen und zurückliefern

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Funktionen</title>
</head>
```

```
<body>
<h1>Die Quadratzahlen</h1>
<script type="text/javascript">

function quadrat(zahl)
{
  var quadrat;

  quadrat = loop * loop;
  return quadrat;
}
var loop;
for (loop=1; loop <= 10; ++loop)
{
  document.write("<p>Das Quadrat von ");
  document.write(loop);
  document.write(" ist ");
  document.write(quadrat(loop));
  document.write("</p>");
}
</script>
</body>
</html>
```

Eine Funktion kann immer nur einen Wert als Ergebniswert zurückliefern. Dies bedeutet aber nicht, dass sie nur eine return-Anweisung enthalten kann. Eine Funktion kann durchaus mehrere return-Anweisungen enthalten - beispielsweise um in Abhängigkeit von einer if-Bedingung verschiedene Ergebniswerte zurückzuliefern.

```
function demo()
{
  ...
  if(Bedingung)
  {
    ...
    return true;
  }
  else
  {
    ...
    return false;
  }
}
```

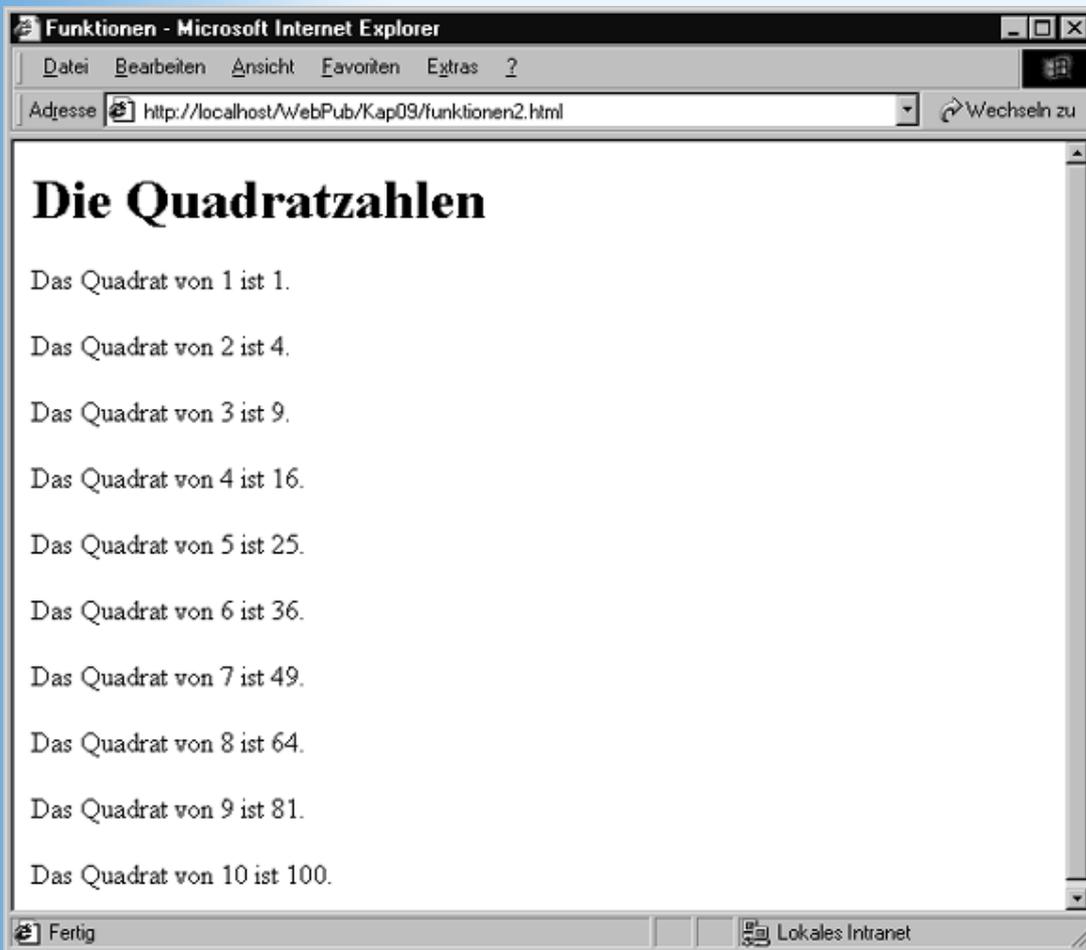


Abbildung 9.6: Mit Hilfe einer Funktion und einer Schleife erzeugte Ausgabe

9.5 Klassen und Objekte

Bisher gehörten alle von uns definierten Variablen dem einen oder anderem Standarddatentyp von JavaScript an (siehe Tabelle 9.3). JavaScript erlaubt aber auch die Definition eigener Datentypen - und zwar in Form von Klassen.

Die Definition von Klassen führt uns zur objektorientierten Programmierung, der bei der modernen Software-Erstellung eine immer größere Bedeutung zukommt (weswegen professionell eingesetzte Programmiersprachen wie C++ oder Java alle objektorientiert sind). Für die Programmierung mit JavaScript ist die objektorientierte Programmierung bei weitem nicht so wichtig. Dies liegt daran, dass JavaScript-Skripte meist so kurz und unkompliziert sind, dass sich die Definition von Klassen selten lohnt. Sie werden daher selten in die Verlegenheit geraten, eigene Klassen zu definieren, und Sie werden auch nur selten über Skripte stolpern, in denen andere JavaScript-Programmierer dies getan haben. Trotzdem lohnt sich die Auseinandersetzung mit den objektorientierten Konzepten, denn wenn wir auch keine eigenen Klassen und Objekte definieren, so müssen wir doch häufig auf die vom Interpreter erstellen Klassen und Objekte zurückgreifen - beispielsweise `window`, `document`, `Math`, etc. Ich sage »müssen«, dabei ist es im Grunde ein Vorteil für uns, dass JavaScript diese Dinge in Form von Klassen und Objekten zur Verfügung stellt, denn die Programmierung wird dadurch logischer und einfacher. Wenn Sie sich erinnern, wie einfach es ist, über `document.write()` einen Text auszugeben oder mit `document.bgColor` die Hintergrundfarbe einer Webseite festzulegen, werden Sie dies sicherlich bestätigen.

Um die Arbeit mit Klassen, Objekten und ihren Methoden und Eigenschaften aber noch besser verstehen zu können, tauchen wir jetzt ein in die Welt der Klassen und Objekte.

Klassen und Objekte

Klassen sind im Grunde Beschreibungen von Dingen, mit denen ein Programm arbeiten möchte. Man kann sich das in etwa wie beim Zinnfigurengießen vorstellen. Nehmen wir an Sie wollen irgendeine historische Schlacht nachstellen. Das Erste, was Sie machen, ist sich für die beteiligten Figuren (Soldaten und Offiziere der beteiligten Länder, Kanonen, Pferde, etc.) Gussformen zu kaufen. Danach gehen Sie daran, die Figuren zu gießen, zu bemalen und aufzustellen. Ranghohe Offiziere werden Sie nur wenige Male gießen, einfache Soldaten oder Reiter werden sie öfters gießen.

Übertragen auf ein JavaScript-Programm entsprechen die Gussformen den Klassen. Wenn Sie ein Programm schreiben würden, das die betreffende Schlacht simuliert, würden Sie für die beteiligten Figuren keine Gussformen erwerben, sondern

Klassen definieren. Und statt aus den Gussformen die eigentlichen Figuren zu gießen, erzeugen Sie in Ihrem Programm die Figuren als *Objekte* der Klassen - einen Vorgang, den man auch als Instanzbildung bezeichnet. Klassen für die ranghöheren Offiziere werden Sie nur wenige Male instantiiieren, während Sie auf der Grundlage der Klassen für Soldaten, Pferde oder Kanonen sicherlich mehrere Objekte erzeugen werden.

Wie die Figuren, die aus einer Form gegossen werden, sind auch die Objekte einer Klasse alle gleich. Die Objekte der Klasse Major sind alles Majore, die Objekte der Klasse Soldat sind alles Soldaten. Die einzelnen Objekte einer Klasse können aber durchaus individuelle Züge aufweisen. Nehmen wir die einfachen Soldaten. Die gegossenen Zinnsoldaten können Sie beispielsweise bemalen, um so den einzelnen Soldatenfiguren Individualität zu verleihen (wobei die Figur immer noch ein Soldat, sprich ein Guss der Form Soldat, bleibt). Die Objekte der Klasse Soldat kann man natürlich nicht bemalen, aber sie verfügen über bestimmte Eigenschaften, die in der Klasse definiert sind (beispielsweise Name, Alter, Haarfarbe). Diesen Eigenschaften kann man individuelle Werte zuweisen.

Klassen definieren

Klassen werden in JavaScript durch Definition einer passenden »Konstrukturfunktion« angelegt. Der Name der Konstrukturfunktion ist auch der Name der Klasse.

Eigenschaften

Im Konstruktor werden die Eigenschaften der Klasse definiert. Allerdings werden die Eigenschaften nicht mit dem Schlüsselwort `var` definiert (wie man vielleicht annehmen könnte), sondern mit Hilfe des Schlüsselworts `this`.



Das Schlüsselwort `this` kann in Konstruktoren- und Methodendefinitionen von Klassen verwendet werden und bezeichnet das jeweils aktuelle Objekt (für das der Konstruktor/die Methode später zur Laufzeit aufgerufen wird).

```
function MeineKlasse()  
{  
  this.eigenschaft1 = wert;  
  ...  
}
```

Dabei wird der Eigenschaft üblicherweise auch gleich ein Wert zugewiesen. Den Wert können Sie direkt angeben (siehe oben). Sie können aber auch im Konstruktor einen Parameter für die Eigenschaft definieren und der Eigenschaft dann den Parameter zuweisen. Letztere Vorgehensweise hat den Vorteil, dass man den Eigenschaften später bei der Erzeugung der Objekte Werte zuweisen kann.

```
function Personalien(name, vorname, alter)  
{  
  this.name = name;  
  this.vorname = vorname;  
  this.alter = alter;  
  this.hobbys = "unbekannt";  
}
```

Methoden

Methoden werden ähnlich wie Eigenschaften definiert. Als Wert weist man ihnen den eigenen Methodennamen zu. Anschließend wird die Methode unterhalb der Konstrukturfunktion definiert.

```
function Personalien(name, vorname, alter)  
{  
  this.name = name;           // Eigenschaft  
  this.vorname = vorname;    // Eigenschaft  
  this.alter = alter;        // Eigenschaft  
  this.hobbys = "unbekannt"; // Eigenschaft  
  this.write = write;        // Methode  
}
```

```
}
function write()           // Methodendefinition
{
  document.write(this.vorname + " ");
  document.write(this.name + ", ");
  document.write(this.alter + ", ");
  document.write("Hobbys: " + this.hobbys);
}
```

Objekte erzeugen

Um Objekte einer Klasse zu erzeugen, benutzt man das Schlüsselwort `new`, ruft den Konstruktor auf und übergibt diesem Werte für die Eigenschaften des Objekts.

```
angestellter = new Personalien("Louis", "Dirk", 34);
```

Auf Klassenelemente zugreifen

Um auf die Eigenschaften oder Methode eines Objekts zuzugreifen, verwendet man den Punkt-Operator. Diesen setzt man zwischen den Objektnamen und den Namen der Eigenschaft oder Methode

```
angestellter.hobbys = "Schach, China, Volleyball";
angestellter.write();
```

with-syntax

Wenn Sie in mehreren, aufeinander folgenden Anweisungen auf die Eigenschaften und/ oder Methoden eines Objekts zugreifen möchten, brauchen Sie nicht immer das Objekt voranzustellen.

Statt

```
angestellter.name = "Hiller";
angestellter.vorname = "Angus";
angestellter.alter = 42;
angestellter.hobbys = "Angeln";
```

können Sie auch schreiben:

```
with (angestellter)
{
  name = "Hiller";
  vorname = "Angus";
  alter = 42;
  hobbys = "Angeln";
}
```

Zugriff innerhalb der Klasse

Will man innerhalb einer Methodendefinition auf andere Elemente (Eigenschaften oder Methoden) der gleichen Klasse zugreifen, braucht man kein vorangehendes Objekt, sondern kann das Element direkt über seinen Namen ansprechen.

Beispiel

In einem zusammenhängenden Beispiel sieht dies dann folgendermaßen aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>

  <title>Klassen</title>
```

```

<meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
// Definition der Klasse Personalien
function Personalien(name, vorname, alter)
{
  this.name = name;           // Eigenschaft
  this.vorname = vorname;     // Eigenschaft
  this.alter = alter;         // Eigenschaft
  this.hobbys = "unbekannt";  // Eigenschaft
  this.write = write;         // Methode
}
function write()              // Methodendefinition
{
  document.write(this.vorname + " ");
  document.write(this.name + ", ");
  document.write(this.alter + ", ");
  document.write("Hobbys: " + this.hobbys);
}
</script>
</head>
<body>
<p>
<script type="text/javascript">
  // Erzeuge angestellter als Objekt der Klasse Personalien
  angestellter = new Personalien("Louis", "Dirk", 34);
  // Ändere die Eigenschaft hobbys
  angestellter.hobbys = "Schach, China, Volleyball";
  // Rufe die Methode write auf
  angestellter.write();
</script>
</p>
</body>
</html>

```



Abbildung 9.7: Ausgabe von `angestellter.write()`

9.6 Vordefinierte Klassen und Objekte

Bei der Programmierung stehen Ihnen eine Reihe von Klassen und Objekten zur Verfügung, die Ihnen zum einen verschiedene nützliche Methoden zur Verfügung stellen, zum anderen als Schnittstelle zu den HTML-Elemente der Webseite dienen. Diese Klassen und Objekte kann man grob in drei Kategorien unterteilen:

- Hilfsklassen, die in der ECMA-Sprachspezifikation für JavaScript definiert sind
- Browser-Objekte, die vom Browser erzeugt und zur Verfügung gestellt werden
- HTML-Objekte, mit denen man auf die HTML-Elemente der Webseite zugreifen kann.

Wie man aus JavaScript-Code heraus auf HTML-Elemente zugreift, erfahren Sie morgen. Zum Ausklang des heutigen Tages begnügen wir uns mit der Vorstellung der wichtigsten Hilfsklassen und Browser-Objekte.

Die Hilfsklassen

Diese Klassen bieten Lösungen für verschiedene, häufig auftretende Programmieraufgaben an.

Die Array-Klasse

Manchmal möchte man in einem Skript mehrere gleichartige Daten verarbeiten. Das kann sehr mühsam sein, wenn man für jeden zu verarbeitenden Wert eine eigene Variable einrichten muss.

Nehmen wir beispielsweise an, Sie führen auf Ihrer Webseite einen Intelligenztest durch. Es gibt insgesamt zwanzig Aufgaben mit jeweils mehreren Unteraufgaben. Insgesamt sind es hundert Unteraufgaben und für jede dieser Unteraufgaben kann man 1 Punkt erhalten. Wir werden dies hier nicht nachprogrammieren, aber damit Sie eine etwas konkretere Vorstellung davon bekommen, möchte ich kurz anskizzieren, wie man so etwas realisieren könnte. Die einzelnen Aufgaben könnte man als Formulare realisieren. Jedes Formular erhält einen Abschicken-Schalter, den der Besucher der Website anklickt, wenn er die Aufgabe mit allen Unteraufgaben bearbeitet hat. Mit dem Abschicken-Schalter ist ein Skript verbunden, das die Aufgaben auswertet, indem es die Punkte für die Unteraufgaben zusammenzählt. Die Punktzahl für die Aufgabe könnte dann in einem hidden-Feld (siehe Kapitel 17.7) des Formulars oder einer globalen JavaScript-Variablen gespeichert werden. Für unser Beispiel nehmen wir an, dass die Werte in globalen JavaScript-Variablen gespeichert werden.

```
var aufgabe1 = 3;
var aufgabe2 = 7;
var aufgabe3 = 7;
...
```

Hat der Besucher der Website den Test vollständig ausgefüllt, sollen die Punkte für die zwanzig Aufgaben zusammengezählt und mit 1.6 multipliziert werden. Das Ergebnis wird dann als erreichter Intelligenzquotient ausgegeben werden.⁵ Als Skript-Code sähe dies dann folgendermaßen aus:

```
var ergebnis = 0;
ergebnis = aufgabe1 + aufgabe2 + aufgabe3 + aufgabe4 + aufgabe5 +
    aufgabe6 + aufgabe7 + aufgabe8 + aufgabe9 + aufgabe10 +
    aufgabe11 + aufgabe12 + aufgabe13 + aufgabe14 + aufgabe15 +
    aufgabe16 + aufgabe17 + aufgabe18 + aufgabe19 + aufgabe20;
ergebnis *= 1.6;
document.write("<p>Erreichter Intelligenzquotient: ");
document.write(ergebnis);
document.write("</p>");
```

Die Aufsummierung sieht ziemlich scheußlich aus. Mit Hilfe eines Arrays (zu deutsch auch »Feld« genannt) könnte man dies viel eleganter lösen.

Zu Anfang der Webseite erzeugt man das Array als globales Objekt:

```
var aufgaben = new Array(20);
```

Unser Array-Objekt heißt aufgaben. Es wird mit dem new-Operator als Objekt der Klasse Array erzeugt. Aus dem vorangehenden Abschnitt wissen Sie, dass die Erzeugung eines Objekts immer mit dem Aufruf des Konstruktors der Klasse verbunden ist. Der Konstruktor der Klasse Array erwartet eine Zahlenangabe, die ihm anzeigt, wie groß das zu erzeugende Array-Objekt sein soll, sprich wie viele Werte in ihm gespeichert werden sollen. Da wir die Punkte aus zwanzig Aufgaben in dem Array verwalten wollen, übergeben wir als Argument 20.

Wir haben jetzt ein Array-Objekt, das aus 20 Elementen (einzelnen Variablen) besteht.



Abbildung 9.8: Elemente in einem Array-Objekt⁶

In den Ereignisbehandlungsfunktionen zu den Abschicken-Schaltern der einzelnen Aufgaben sollen nun den Elementen im Array die erreichten Punktzahlen zugewiesen werden. Die Frage ist nur, wie man auf die einzelnen Elemente im Array zugreifen kann? Nun, zwar haben die Elemente im Array keine eigenen Namen, dafür aber liegen sie nebeneinander in einer festen Reihenfolge im Array.

Man kann daher auf die einzelnen Elemente über einen Index, eine Positionsangabe, zugreifen. Per Definition hat das erste Element im Array dabei den Index 0. Folglich hat das letzte Element in einem Array von 20 Elementen den Index 19. Zum Abspeichern der erreichten Punkte in den Elementen des Arrays schreiben wir also:

```
aufgaben[0] = 3;
aufgaben[1] = 3;
aufgaben[2] = 3;
...
```

Soweit sieht man noch keinen großen Unterschied zwischen der Programmierung mit einem Array oder einer Reihe von normalen Variablen. Wenn wir allerdings daran gehen, die Werte aus den Elementen aufzuaddieren, erweist sich der indizierte Zugriff auf die Array-Elemente als extrem hilfreich.

```
var summe = 0;
var loop;
for (loop = 0; loop < 20; ++loop)
{
    summe += aufgaben[loop];
}
```

Der Trick ist, eine Schleife aufzusetzen und die Schleifenvariable als Index für die Array-Elemente zu verwenden.

Das Durchlaufen der Elemente eines Arrays mit Hilfe einer Schleife ist eine so typische und häufig benötigte Konstruktion, dass es dafür sogar eine besondere for-Syntax gibt:

```
for (loop in aufgaben)
{
    summe += aufgaben[loop];
}
```

Diese Schleife durchläuft automatisch nacheinander alle Elemente im Array. Man braucht sich noch nicht einmal um die korrekte Initialisierung und Überprüfung der Schleifenvariable zu kümmern.

Was kann man noch mit Arrays machen? Tabelle 9.11 gibt Ihnen eine Übersicht.

Operation	Beschreibung
Länge bestimmen	Die Eigenschaft length enthält die Anzahl Elemente im Array. <pre>var feld = new Array(10); var anzahl = feld.length; // = 10</pre>
Elemente ausgeben	Mit Hilfe der Methode join() kann man einen Textstring erzeugen, der alle Elemente im Array enthält. Als Argument kann man der Methode eine Zeichenkombination übergeben, die zum Trennen der Elemente im String verwendet werden soll. <pre>var ausgabe = feld.join(", "); document.write(ausgabe);</pre>
Reihenfolge umkehren	Mit Hilfe der Methode reverse() kann man die Reihenfolge der Elemente im Array umkehren. <pre>feld.reverse();</pre>

<p>Array sortieren</p>	<p>Mit Hilfe der Methode <code>sort()</code> kann man die Elemente im Array sortieren.</p> <pre>feld.sort();</pre> <p>Standardmäßig interpretiert die Methode alle Elemente als Strings und sortiert sie lexikographisch. Will man Zahlen als echte Zahlen sortieren, muss man eine passende Vergleichsfunktion aufsetzen, die festlegt, wie je zwei Elemente im Array zu vergleichen sind:</p> <pre>function vergleiche(a, b) { if (a < b) return -1; if (a > b) return 1; if (a == b) return 0; }</pre> <p>Diese Funktion übergibt man dann <code>sort()</code> beim Aufruf:</p> <pre>feld.sort(vergleiche);</pre>
<p>Dynamische Arrays</p>	<p>Arrays können dynamisch wachsen. Sie brauchen dazu nur ein Element mit einem höheren Index zu definieren.</p> <p>Nehmen wir an, Sie hätten folgendes Array definiert:</p> <pre>var feld = new Array(10);</pre> <p>Um dieses Array um fünf weitere Elemente zu erweitern, brauchen Sie nur zu schreiben:</p> <pre>feld[14] = 0;</pre> <p>Beachten Sie aber, dass die Elemente mit den Indizes 10, 11, 12 und 13 danach undefinierte Werte haben - bevor Sie mit diesen Elementen arbeiten können, müssen Sie Ihnen richtige Werte zuweisen.</p>
<p>Strings als Indizes</p>	<p>Sie können auch Strings als Indizes verwenden. Dazu müssen Sie ein leeres Array definieren:</p> <pre>var feld = new Array();</pre> <p>Danach nehmen Sie nacheinander Elemente in das Array auf, wobei Sie statt Positionszahlen eindeutige Strings zur Indizierung der Elemente verwenden:</p> <pre>feld["Rainer"] = "3986751"; feld["Otto"] = "555312"; feld["Maria"] = "719903";</pre>

Tabelle 9.11: Array-Operationen

Zum Abschluss schauen wir uns noch ein kleines Beispielskript an. Mit Hilfe des folgenden Skriptes kann der Besucher der Website den Mittelwert einer Zahlenreihe berechnen (beispielsweise das Temperaturmittel für den letzten Monat, oder seinen Notendurchschnitt, etc.)

Listing 9.12: arrays.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Arrays</title>
```

```

</head>
<body>
<h1>Arrays</h1>
<script type="text/javascript">
  // Leeres Array-Objekt definieren
  var werte = new Array();
  // Werte dynamisch einlesen
  var eingabe, loop = 0;
  var weiter = true;
  while (weiter == true)
  {
    eingabe = prompt("Geben Sie einen Wert ein ('ende' zum Beenden): ",
                    "");
    if (eingabe == "ende")
      weiter = false;
    else
    {
      werte[loop] = parseFloat(eingabe);
      ++loop;
    }
  }
  // Summe berechnen und Mittelwert ausgeben
  var summe = 0;
  for (loop in werte)
  {
    document.write("<p>Wert: "); // einzelne Werte zur
    document.write(werte[loop]); // Kontrolle ausgeben
    document.write("</p>");
    summe += werte[loop]; // aufsummieren
  }
  document.write("<p>Mittelwert: ");
  document.write(summe / werte.length);
  document.write("</p>");
</script>
</body>
</html>

```

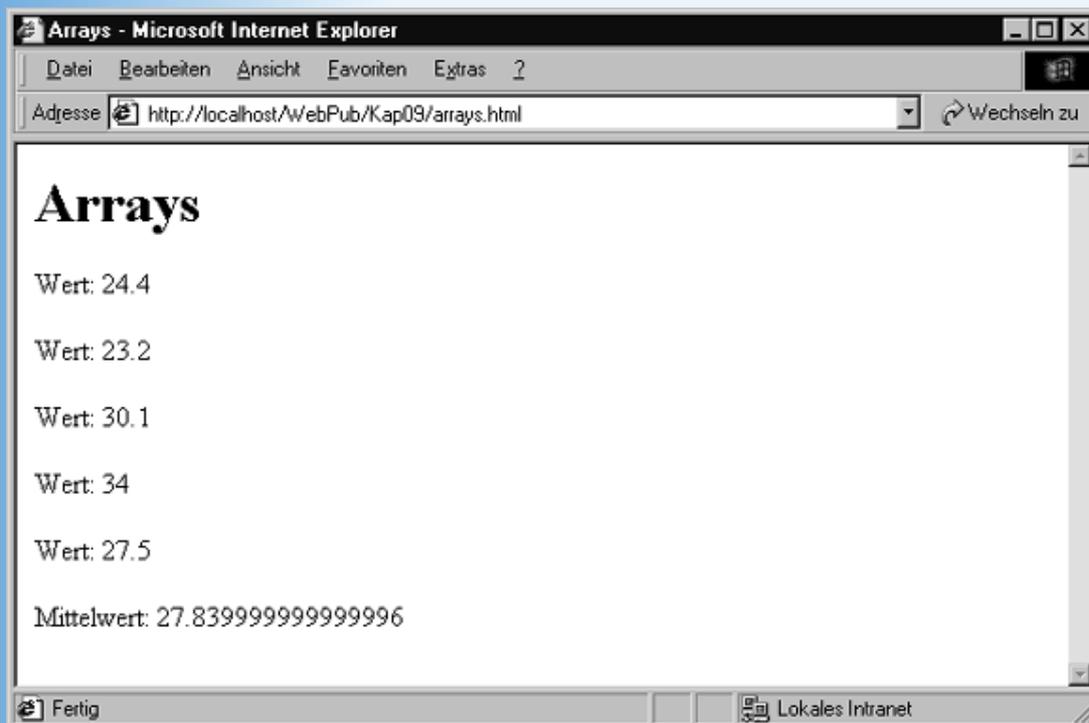


Abbildung 9.9: Mittelwertberechnung mit Hilfe eines Arrays

Die Date-Klasse

Objekte der Klasse Date repräsentieren Datums- und Zeitangaben.

Ruft man den Konstruktor ohne Parameter auf, wird das Objekt mit dem aktuellen Datum und der augenblicklichen Uhrzeit initialisiert (wozu auf die Systemzeit des Rechners zurückgegriffen wird).

```
datum = new Date();
```

Will man ein Date-Objekt für einen speziellen Tag (Uhrzeit) erzeugen, muss man die entsprechenden Daten an den Konstruktor übergeben:

```
datum = new Date(jahr, monat, tag);
datum = new Date(y, m, t, std, min, sec);
```

Welche Werte Sie für Jahr, Monat, etc. eingeben müssen, können Sie der folgenden Tabelle entnehmen.

Angabe	Werte
Jahr	Vergangene Jahre seit 1900
Monat	0 - 11
Tag	1 - 31
Stunde	0 - 23
Minute	0 - 59
Sekunde	0 - 59

Tabelle 9.12: Zeitangaben für Date-Konstruktor

Beispiel:

```
geburtstag = new Date(62, 7, 11); // 11. August 1962
```

Zum Abfragen und Verändern des Datums stehen folgende Methoden zur Verfügung.

Methode	Beschreibung
getDate()	Liefert Tag im Monat
getDay()	Liefert Wochentag
getHours()	Liefert Stunden
getMinutes()	Liefert Minuten
getMonth()	Liefert Monat
getSeconds()	Liefert Sekunden
getYear()	Liefert Jahr
setDate(Tag)	Setzt den Tag
setHours(Stunden)	Setzt die Stunden
setMinutes(Minuten)	Setzt die Minuten
setMonth(Monat)	Setzt den Monat
setSeconds(Sekunden)	Setzt die Sekunden
setYear(Jahr)	Setzt das Jahr

getTime()	Liefert Anzahl der Millisekunden, die zwischen dem 1.1.1970 0:00 Uhr und dem im Date-Objekt gespeicherten Datum vergangen sind. Mit Hilfe dieser Methode kann man auch die Differenz zwischen zwei Datumsangaben berechnen. <pre>var diff = date1.getTime() - date2.getTime();</pre>
getTimeZoneoffset()	Zeitverschiebung (in Minuten) gegenüber Greenwich Mean Time.
toGMTString()	Wandelt Date-Objekt als GMT-Datum in String um.
toLocaleString()	Wandelt Date-Objekt nach lokaler Zeit in String um.
UTC(j, m, t) UTC(j, m, t, std, min, sec)	Liefert Anzahl der Millisekunden, die zwischen dem 1.1.1970 0:00 Uhr und dem im Date-Objekt gespeicherten Datum gemäß Greenwich Mean Time vergangen sind.

Tabelle 9.13: Date-Methoden

Die Math-Klasse

Diese Klasse stellt uns eine Reihe von Konstanten (E, LN2, LN10, LOG2E, LOG10E, PI, SQRT1_2, SQRT2) und mathematischen Funktionen (siehe Tabelle) zur Verfügung.

Methode	Beschreibung
abs(x)	Absolutwert
acos(x)	Arkuskosinus
asin(x)	Arkussinus
atan(x)	Arkustangens
atan2(gk, ak)	Arkustangens (für Längen der Gegen- und Ankathete)
ceil(x)	Nächsthöhere Ganzzahl
cos(x)	Kosinus
exp(x)	Exponentialfunktion
floor(x)	Nächstniedrigere Ganzzahl
log(x)	Logarithmus zur Basis e
max(x, y)	Maximum zweier Werte
min(x, y)	Minimum zweier Werte
pow(x, y)	Potenzfunktion (x hoch y)
random(x)	Zufallszahl zwischen 0 und 1, siehe auch Listing 9.10
round(x)	Runden
sin(x)	Sinus
sqrt(x)	Wurzel
tan(x)	Tangens

Tabelle 9.14: Date-Methoden



Sie auch Listing 9.5.

Die String-Klasse

Objekte der Klasse String kann man durch Aufruf des Konstruktors oder durch einfache Zuweisung eines Strings an eine Variable erzeugen.

```
var text = new String();  
var text = new String("Hallo");  
var text = "Hallo";
```

Dass Strings in JavaScript immer Objekte der Klasse String sind, ist von großem Vorteil für uns, denn neben dem +-Operator zum Aneinanderreihen von Strings stellt uns die Klasse String eine Reihe von Methoden zur Bearbeitung von Strings zur Verfügung.

Methoden	Beschreibung
charAt(position)	Liefert Zeichen an bestimmter Position in String zurück. Das erste Zeichen steht an der Position 0.
charCodeAt(position)	Liefert den UNICODE-Wert des Zeichens an der angegebenen Position im String zurück. Das erste Zeichen steht an der Position 0.
concat(string1, string2, ...)	Reiht Strings aneinander (wie +-Operator)
indexOf(suchString, pos)	Durchsucht einen String nach einem Teilstring und liefert die Anfangsposition des gesuchten Teilstrings im String zurück. Die Suche beginnt ab der Position pos.
lastIndexOf(suchString, pos)	Funktioniert wie indexOf, durchsucht den String aber von hinten.
localeCompare(string)	Vergleicht den übergebenen String mit dem eigenen String unter Berücksichtigung lokaler Eigenheiten (Umlaute, etc.) Die Rückgabewerte sind implementierungsspezifisch.
match(regExp)	Sucht im String nach Teilstring, die dem regulären Ausdruck regExp entsprechen. Liefert Array der gefundenen Vorkommen zurück.
replace(suchString, ersetzenString)	Sucht im String nach dem Teilstring suchString und ersetzt ihn durch ersetzenString.
search(regExp)	Sucht im String nach Teilstring, die dem regulären Ausdruck regExp entsprechen. Liefert Anfangsposition des gefundenen Teilstrings zurück.
slice(start, ende)	Extrahiert den String zwischen den Positionen start und ende.
split(trennzeichen)	Teilt den String bei jedem Vorkommen des angegebenen Trennzeichens und liefert die Teilstrings als Array zurück.
substring(start, ende)	Liefert den Teilstring zwischen den Positionen start und ende zurück.
toLowerCase()	Umwandlung in Kleinbuchstaben.
toLocaleLowerCase()	Umwandlung in Kleinbuchstaben (unter Berücksichtigung nationaler Eigenheiten)
toUpperCase()	Umwandlung in Großbuchstaben.
toLocaleUpperCase()	Umwandlung in Großbuchstaben (unter Berücksichtigung nationaler Eigenheiten)

Tabelle 9.15: String-Methoden

Als Bonus gibt es noch eine Eigenschaft: length, die Länge des Strings in Zeichen.

Weitere Hilfsklassen

Klasse	Beschreibung
Boolean	Zur expliziten Erzeugung von Booleschen Objekten: <pre>var weitermachen = new Boolean(true);</pre>

Function	Alternative Syntax zur Erzeugung einfacher Funktionen: <code>meineFunkt = new Function("x", "y", "return x*y");</code>
Number	Zur expliziten Erzeugung von Zahlen-Objekten: <code>var zahl = new Number(123);</code>
RegExp	Klasse zur Erzeugung regulärer Ausdrücke. Reguläre Ausdrücke sind ein sehr mächtiges Hilfsmittel zum Aufspüren von Textstellen in Strings. Obwohl die Arbeit mit regulären Ausdrücken nicht allzu kompliziert ist (wenn Sie schon einmal Ihr Dateisystem mit *.html nach HTML-Dateien durchsucht haben, haben Sie schon mit regulären Ausdrücken gearbeitet), ist das Thema doch zu umfangreich, um im Rahmen dieses Buches ausführlich behandelt zu werden.
Error	Klasse zur Erzeugung von Fehlerobjekten

Tabelle 9.16: Sonstige JavaScript-Hilfsklassen

Die Browser-Objekte

Die folgenden Objekte werden vom Browser erzeugt und stehen nach dem Laden der Webseite zur Verfügung.

Das window-Objekt

Dieses Objekt repräsentiert das Browserfenster. Abgesehen davon, dass es das übergeordnete Objekt zu den document-, location- und history-Objekten ist (diese sind Eigenschaften des window-Objekts) stellt es uns eine Reihe nützlicher Funktionen zur Verfügung.

Methode	Beschreibung
open(url, name, eigenschaften)	Öffnet ein neues Browserfenster (siehe Kapitel 8.3.5) Im String eigenschaften können Sie folgende Angaben machen: <code>"toolbar=yes,scrollbars=no,menubar=no,status=yes, resizable=yes,location=no,directories=no,width=400, height=300"</code>
close()	Schließt das Browserfenster
alert(text)	Ruft ein Meldungsfenster auf (siehe Kapitel 8.3.3)
confirm(text)	Ruft ein Ok/Abbrechen-Dialog auf
prompt(text, vorgebe)	Ruft eine Eingabeaufforderung auf (siehe Übung 1)
setTimeout(funktion, n)	Ruft die angegebene Funktion nach n Millisekunden auf.

Tabelle 9.17: window-Methoden

Das document-Objekt

Dieses Objekt repräsentiert die Webseite. Es enthält eine Reihe von allgemein unterstützten Methoden und Eigenschaften.

Methode	Beschreibung
clear()	Löscht Webseite.
close()	Schließt Webseite. Danach sind keine Ausgaben mit write/writeln mehr möglich.
open()	Öffnet eine Webseite.

write(string)	Ausgabe eines Textes in Webseite.
writeln(string)	Ausgabe eines Textes in Webseite (mit abschließendem Zeilenumbruch)

Tabelle 9.18: document-Methoden

Eigenschaft	Beschreibung
alinkColor	Farbe aktivierter Links
anchors[]	Array, über das man auf die Anker-Elemente der Webseite zugreifen kann
applets[]	Array, über das man auf die Applet-Elemente der Webseite zugreifen kann
bgColor	Hintergrundfarbe
cookie	Von der Webseite gesetzte Cookies
domain	Domänenname des Webservers, von dem die Webseite stammt
fgColor	Vordergrundfarbe
forms[]	Array, über das man auf die Formulare der Webseite zugreifen kann
images[]	Array, über das man auf die -Elemente der Webseite zugreifen kann
lastModified	String, der das Datum der letzten Änderung der Webseite enthält
linkColor	Farbe von Hyperlinks
links[]	Array, über das man auf die Hyperlinks der Webseite zugreifen kann
location	Webadresse der Webseite
referrer	URL, von dem der Besucher zu dieser Seite gekommen ist
title	Titel der Webseite
URL	URL der Webseite
vlinkColor	Farbe besuchter Hyperlinks

Tabelle 9.19: document-Eigenschaften



Manche Eigenschaften, beispielsweise die Vordergrundfarbe, können nach dem Laden der Seite nicht mehr geändert werden (sie sind praktisch nur bei der dynamischen Erzeugung neuer Webseiten interessant, siehe Kapitel 8.3.5).

Das location-Objekt

Dieses Objekt enthält Informationen über den Header des Dokuments.

Methode	Beschreibung
reload()	Seite nochmals laden.
replace(URL)	Lädt neue Webseite und löscht die alte Webseite aus der History.

Tabelle 9.20: location-Methoden

Eigenschaft	Beschreibung

hash	Anker eines Hyperlinks
host	Host und Port
hostname	Name des Servers
href	Gesamte URL
pathname	Pfad und Dateiname
port	Portnummer
protocol	Verwendetes Protokoll
search	Eventuell an URL angehängter Suchstring (zur Verarbeitung in CGI-Programmen)

Tabelle 9.21: location-Eigenschaften

Über das location-Objekt können Sie sich über die Herkunft einer Webseite informieren oder eine neue Webseite laden:

```
function surfe_zu()
{
  location.href = "http://www.w3.org/";
}
```

Das history-Objekt

Liste der URLs, die der Besucher innerhalb des aktuellen Fensters besucht hat.

Methode	Beschreibung
back()	Wechselt zu der Webseite, die in der History-Liste vorangeht.
forward()	Wechselt zu der Webseite, die in der History-Liste nachfolgt.
go(x)	Wechselt zu der Webseite, die x Positionen in der History-Liste entfernt ist. x kann ein negativer oder positiver ganzzahliger Wert sein.

Tabelle 9.22: history-Methoden

Eigenschaft	Beschreibung
length	Anzahl der Einträge in History-Liste

Tabelle 9.23: history-Eigenschaften

Mit Hilfe des history-Objekts können Sie nicht nur zu anderen Webseiten wechseln, Sie können auch die aktuelle Webseite neu laden lassen:

```
function aktualisiere()
{
  history.go(0);
}
```

Das navigator-Objekt

Über dieses Objekt können Sie sich über den vom Besucher Ihrer Webseite verwendeten Browser informieren.

Methode	Beschreibung
javaEnabled()	Liefert true, wenn der Browser Java unterstützt.

Tabelle 9.24: navigator-Methoden

Eigenschaft	Beschreibung
appName	Codename des Browsers
appVersion	Name des Browsers
mimeTypes[]	Version des Browsers
platform	Vom Browser unterstützte MIME-Typen
plugins[]	Plattform, für die der Browser erstellt wurde
userAgent	Array der installierten Plug-Ins
	Kombination von appName und appVersion-Informationen

Tabelle 9.25: navigator-Eigenschaften

Die Abfrage des verwendeten Browsers (und möglicherweise auch der Browser-Version) ist vor allem dann wichtig, wenn Sie JavaScript-Techniken verwenden, die in den verschiedenen Browsern auf verschiedene Weise durchgeführt werden. Am morgigen Tag werden wir auf diesen Themenbereich ausführlich eingehen.

9.7 Zusammenfassung

Heute haben wir uns mit der JavaScript-Syntax beschäftigt. Für einige Leser wird dies nicht nur der Einstieg in JavaScript, sondern auch der Einstieg in die Programmierung überhaupt gewesen sein. Deshalb sollten wir kurz die wichtigsten Konzepte der Programmierung mit höheren Programmiersprachen wiederholen. (Wenn es auch - insbesondere im Bereich der Künstlichen Intelligenz - Programmiersprachen gibt, die auf anderen Konzepten aufbauen, kann man die an diesem Tag vorgestellten Konzepte in vielen aktuellen Programmiersprachen wiedererkennen.)

Programme dienen allgemein der Datenverarbeitung. Diese Daten können in einem Programm in Form von Konstanten oder Variablen repräsentiert werden.

Variablen sind letztendlich Aliase für Speicherbereiche, in denen man Werte (Daten) abspeichern kann. Nach Art und Aufbau der Daten unterscheidet man verschiedene Datentypen: Boolesche Daten, Ganzzahlen, Fließkommazahlen, Strings. Im Gegensatz zu streng typisierten Programmiersprachen wie C++ oder Java, die für jeden dieser Datentypen eigene Variablentypen kennen, gibt es in JavaScript nur einen Art von Variablen, der man Werte jedes beliebigen Datentyps zuweisen kann. Trotzdem unterscheidet auch JavaScript zwischen den verschiedenen Datentypen. So kann man beispielsweise am Format einer Konstanten ablesen, von welchem Datentyp sie ist. Zudem sind die meisten Operatoren datentypspezifisch.

Mit Hilfe von Operatoren kann man Daten verarbeiten: Man kann sie an Variablen zuweisen, man kann aus ihnen neue Daten berechnen, man kann sie vergleichen.

Vergleiche von Daten werden meist in Verbindung mit Verzweigungen oder Schleifen ausgeführt. Dabei geht es darum, je nach dem Ergebnis eines Vergleichs den einen oder anderen Anweisungsblock auszuführen (Verzweigung) beziehungsweise einen Anweisungsblock mehrfach ausführen zu lassen (Schleife).

Um Code besser strukturieren zu können, verwendet man Funktionen. Ist man gezwungen, eine bestimmte Aufgabe an mehreren Stellen in einem Programm auszuführen (beispielsweise Berechnung des Ergebnisses einer Formel), ist es besser, den Code einmalig in Form einer Funktion zu implementieren und dann die Funktion aufzurufen statt jedes Mal, wo die Berechnung benötigt wird, den Code einzukopieren. (Letzteres Verfahren ist aufwendiger, fehleranfälliger und wenig wartungsfreundlich.) Über Parameter können Funktionen Daten übernehmen, über ihren Rückgabewert können die Funktionen Daten an den aufrufenden Code zurückliefern.

Möchte man komplexere Daten als nur einzelne Zahlen oder Strings verarbeiten, definiert man Klassen. Klassen bestehen

aus einer Reihe von Daten (Eigenschaften genannt), bei denen es sich um Daten der elementaren Datentypen oder um Objekte anderer Klassen handeln kann, und klassenspezifischen Funktionen (Methoden genannt), die gewissermaßen die Operatoren darstellen, über die man die Objekte der Klasse bearbeiten kann. (In manchen Sprachen, beispielsweise C++, kann man auch klassenspezifische Implementierungen für Operatoren definieren.) »Werte« vom Typ einer Klassen bezeichnet man als Objekte.

JavaScript definiert eine Reihe von nützlichen Klassen für viele häufig auftretende Programmierprobleme (Array, String, Math, RegExp). JavaScript-fähige Browser stellen eine Reihe von Objekten für die Skriptprogrammierung zur Verfügung: window, document, navigator, location, history.

9.8 Fragen und Antworten

Frage:

Ich habe schon JavaScript-Variablen gesehen, die ein Dollarzeichen enthielten. Sollte das nicht eigentlich ein Fehler sein?

Antwort:

Nein, seit JavaScript 1.1 ist neben dem Unterstrich auch das Dollarzeichen in Bezeichnern erlaubt. Das Dollarzeichen sollte aber laut ECMA-Standard möglichst nicht von Programmierern verwendet werden. Bedenken Sie auch, dass ältere JavaScript-Interpreter das Dollarzeichen in Bezeichnern nicht akzeptieren.

Frage:

Wenn ein unbekannter Bezeichner auf der linken Seite einer Zuweisung auftaucht, erzeugt der Interpreter automatisch eine neue Variable für den Bezeichner. Was aber passiert, wenn ein unbekannter Bezeichner in einem Ausdruck (also als R-Wert) auftaucht?

Antwort:

In so einem Fall brechen die meisten Browser die JavaScript-Bearbeitung ab.

Frage:

Wenn man die Anweisungen zu einer case-Marke nicht mit einer break-Anweisung abschließt, werden auch die nachfolgenden Anweisungen ausgeführt. Meist ist dies unerwünscht, weswegen wir die Anweisungen zu den case-Marken grundsätzlich mit break; abschließen. Aber vielleicht könnte man dieses »Durchfallen« von case-Marke zu case-Marke ja auch irgendwie sinnvoll nutzen?

Antwort:

Ja, wenn Sie beispielsweise für die Werte 1, 2, 3 und 4 den jeweils gleichen Code ausführen wollen, können Sie schreiben:

```
switch(variable)
{
  case 1:
  case 2:
  case 3:
  case 4: // tue etwas; break;
  case 5: // tue etwas anderes;
}
```

Frage:

Ich kann das Beispiel mit der switch-Verzweigung nicht nachvollziehen. Mein Browser führt sie nicht aus!

Antwort:

In früheren JavaScript-Versionen gab es noch keine switch-Verzweigung. Entsprechend wird die switch-Verzweigung nicht von älteren Browsern unterstützt. Sollten Sie einen neueren Browser verwenden, prüfen Sie, ob Sie nicht vielleicht doch irgendwo einen Tippfehler eingebaut haben.

Frage:

Kann ich in einem Array Werte unterschiedlicher Datentypen speichern, beispielsweise Zahlen und Strings?

Antwort:

Ja, das geht, aber man sollte es möglichst vermeiden. Der Hauptvorteil der Arrays ist, dass man die Elemente im Array in einer Schleife durchlaufen und bearbeiten kann. Das geht aber nicht mehr, wenn im Array ganz unterschiedliche Daten abgespeichert sind, die nicht in dergleichen Weise verarbeitet werden können.

9.9 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

Quiz

2. Welche der folgenden Variablennamen sind keine gültigen Bezeichner?

```
var hallo;  
var drei!;  
var 3fach;  
var Hallo;  
var guter-Wert;  
var Gärung;  
var new;  
var newWert;
```

3. Für welche Werte stehen die folgenden Ausdrücke, wenn wert1 = 13, wert2 = 4, wert3 = 1.5, wert4 = "23"?

```
wert1 / wert2;           // 3.25  
wert1 % wert2;          // 1  
wert1 / wert3;          // 8.6666  
wert1 + wert4 * wert3;   // 47.5  
(wert1 + wert4) * wert3; // 1984.5
```

4. Was ist falsch an der folgenden Schleife?

```
var loop = 1;  
var ergebnis = 0;  
while(loop <= 10)  
{  
    ergebnis = loop*loop*loop;  
    if (ergebnis > 700)  
        continue;  
    document.writeln(ergebnis);  
    ++loop;  
}
```

5. Sie wollen eine Reihe von Messwerten verarbeiten. Welche Datenstruktur wählen Sie zur Verwaltung der Messwerte in Ihrem Skriptcode?
6. Sie wollen zusammengehörende Daten unterschiedlicher Datentypen verarbeiten (beispielsweise Personendaten: Name, Vorname, Alter, Haarfarbe). Welche Datenstruktur wählen Sie zur Verwaltung der Adressen in Ihrem Skriptcode?

Übungen

1. Implementieren Sie ein JavaScript, das beim Aufruf der Webseite ein Passwort abfragt und mit Hilfe einer if-Bedingung sicherstellt, dass nur ausgewiesene Besucher den Inhalt der Webseite zu sehen bekommen.
2. Wandeln Sie die folgende for-Schleife in eine while-Schleife um.

```
// Umrechnung Fahrenheit in Celsius  
var fahrenheit = 0;  
var celsius = 0;  
for (fahrenheit = -30; fahrenheit <= 90; fahrenheit += 10)  
{  
    celsius = (fahrenheit - 32) * 5 / 9;  
    document.write("<p>" + fahrenheit + " Fahrenheit entsprechen "  
        + celsius + " Grad Celsius</p>");  
}
```

}

3. Schreiben Sie Listing 9.10 so um, dass in dem Skriptcode vollkommen zufällige RGB- Farben ausgewählt werden.

1 Programmieranfänger haben im Gegenteil eher Schwierigkeiten damit, zwischen dem String "123" und der Zahl 123 zu unterscheiden.

2 Das »Addieren« von Strings sieht so aus, dass die String einfach aneinandergehängt werden. Man bezeichnet dies auch als Konkatenation.

3 Man denke beispielsweise an die Implementierung von Zählern.

4 Streng genommen, handelt es sich bei den aufgeführten Funktionen um Methoden. Methoden sind aber letztlich nichts anderes als klassenspezifische Funktionen (siehe Abschnitt 9.5).

5 Ich weiß nicht, ob dies eine statthafte Methode zur Berechnung des Intelligenzquotienten ist, aber es reicht für unser Beispiel.

6 Die Elemente eines Arrays müssen im Speicher nicht wirklich wie in der Abbildung dargestellt nebeneinander liegen. Es ist jedoch ein gutes Modell, um sich die Funktionsweise eines Arrays zu veranschaulichen. (Und in manchen Programmiersprachen, wie beispielsweise C/C++ liegen die Elemente von Arrays tatsächlich immer nebeneinander im Speicher.)

Tag 10

Mit JavaScript auf HTML-Elemente zugreifen

Heute werden wir uns anschauen, wie man aus einem JavaScript-Skript heraus auf die HTML-Elemente einer Webseite zugreift. Nebenbei werden wir uns etwas ausführlicher mit der Ereignisbehandlung unter JavaScript beschäftigen, das DOM-Modell zur Repräsentation von Webseiten kennen lernen und uns einige Lösungen für browserunabhängige Webseiten anschauen.

Die Themen im Einzelnen:

- Ereignisbehandlung in HTML und JavaScript
- Welche Ereignisse gibt es?
- Wie behandelt man Klickereignisse für Bilder?
- Das Ereignis-Objekt
- Ereignisse simulieren
- Event-Capturing
- Das DOM-Modell
- Browserunabhängige Implementierungen
- Schaltflächen mit Rollover-Effekt
- Dynamische Listen

10.1 Ereignisse

Wie man Funktionen zur Bearbeitung von HTML-Ereignissen einrichtet, haben wir bereits in Kapitel 8 gesehen. Rekapitulieren wir noch einmal die wichtigsten Schritte.

Das Grundmodell der Ereignisbehandlung

Nehmen wir an, Sie wollen das Anklicken eines Bildes mit dem Aufruf eines Meldungsfensters verbinden. Im Meldungsfenster weisen Sie den Websurfer daraufhin, dass er das Bild nicht weiter anzuklicken braucht, da es nicht mit einem Hyperlink verbunden ist.

1. Sie setzen im Header-Bereich die Funktion zur Bearbeitung des Ereignisses auf.

```
<script type="text/javascript">
function hinweis()
{
    alert("Dieses Bild ist nicht mit einem Hyperlink verbunden!","");
}
</script>
```

2. Sie erweitern das -Tag um das Ereignis-Attribut onclick und weisen diesem die im ersten Schritt aufgesetzte Funktion als »Wert« zu.

```
<body>
```

```

...

...
</body>

```



Besteht der Code zur Behandlung eines Ereignisses nur aus ein oder zwei Anweisungen (wie im obigen Beispiel) lohnt sich die Definition einer Ereignisbehandlungsfunktion im Grunde nicht. In so einem Fall kann man sich überlegen, ob man den Code nicht einfach direkt als Wert des Ereignis-Attributs angibt.

Wenn Sie obiges Beispiel nachprogrammieren und austesten, werden Sie feststellen, dass der Code im Internet Explorer und Netscape 6-Browser wie gewünscht funktioniert, nicht aber im Netscape Navigator 4. Der Grund hierfür ist, dass der Netscape Navigator 4 das onclick-Ereignis nicht in Kombination mit -Elementen unterstützt.

Dies führt uns zu der Frage, welche Ereignisse es überhaupt gibt, für welche HTML- Elemente sie definiert sind und inwieweit sie von den verschiedenen Browsern unterstützt werden.

Die verschiedenen HTML-Ereignisse

In der HTML 4-Spezifikation sind folgende Ereignisse definiert:

Attribut	Ereignis	HTML-Elemente
onblur	Tritt ein, wenn das Element den Fokus verliert. (Ein Element verliert den Fokus, wenn ein anderes Element den Fokus erhält, siehe onfocus)	a, area, label, input, select, textarea, button
onchange	Tritt ein, wenn das Element den Fokus verliert und sich sein Inhalt seit dem Erhalt des Fokus verändert hat.	input, select, textarea
onclick	Tritt ein, wenn der Besucher auf das Element klickt.	Gilt für die meisten Elemente
ondblclick	Tritt ein, wenn der Besucher doppelt auf das Element klickt.	Gilt für die meisten Elemente
onfocus	Tritt ein, wenn das Element den Fokus erhält (aktiviert wird). (Ein Element erhält den Fokus, wenn es angeklickt wird oder der Fokus mit der (Tab)-Taste weitergereicht wird.)	a, area, label, input, select, textarea, button
onkeydown	Tritt ein, wenn das Element aktiviert (angeklickt) ist und der Besucher eine Taste drückt.	Gilt für die meisten Elemente

onkeypress	Tritt ein, wenn das Element aktiviert (angeklickt) ist und der Besucher eine Taste drückt und loslässt.	Gilt für die meisten Elemente
onkeyup	Tritt ein, wenn das Element aktiviert (angeklickt) ist und der Besucher eine Taste loslässt.	Gilt für die meisten Elemente
onload	Tritt ein, nachdem eine Webseite (Fenster) oder alle Frames einer Frameseite geladen wurden	body, frameset
onmousedown	Tritt ein, wenn der Besucher über dem Element eine Maustaste drückt.	Gilt für die meisten Elemente
onmousemove	Tritt ein, wenn der Besucher die Maus über dem Element bewegt.	Gilt für die meisten Elemente
onmouseout	Tritt ein, wenn die Maus aus dem Bereich des Elements herausbewegt wird.	Gilt für die meisten Elemente
onmouseover	Tritt ein, wenn die Maus über einem Element steht.	Gilt für die meisten Elemente
onmouseup	Tritt ein, wenn der Besucher über dem Element eine Maustaste loslässt.	Gilt für die meisten Elemente
onreset	Tritt ein, wenn das Formular zurückgesetzt wurde	form
onselect	Tritt ein, wenn der Besucher in einem Textfeld eine Textpassage markiert	input, textarea
onsubmit	Tritt ein, wenn das Formular abgeschickt wurde	form
onunload	Tritt ein, wenn eine Webseite aus einem Fenster oder einem Frame entfernt wird	body, frameset

Tabelle 10.1: Die HTML/Javascript-Ereignisse

Obige Liste darf man aber bestenfalls als Orientierungshilfe und Richtlinie ansehen. Mehr noch als beim Aufsetzen von HTML-Code gilt bei der JavaScript-Programmierung, dass für den Webautor letzten Endes nicht die Spezifikation, sondern die reale Unterstützung durch die Browser entscheidend ist - und hier gibt es bezüglich der unterstützten Ereignisse erhebliche Abweichungen.

- Die beste und weitreichendste Unterstützung der in der HTML-4-Spezifikation definierten Ereignisse findet man derzeit im Internet Explorer 5 und im Netscape 6- Browser.
- Am restriktivsten ist der Netscape Navigator 4, der die Ereignisbehandlung weitgehend auf Ereignisse von Formularen und deren Steuerelemente sowie auf globale Fensterereignisse und Links beschränkt.
- Auf der anderen Seite gibt es Ereignisse, die von aktuellen Browsern unterstützt werden, die aber nicht im HTML-Standard definiert sind (und es vielleicht auch nie sein werden) - beispielsweise das onabort-Ereignis für das -Tag, das ausgelöst wird, wenn der Websurfer das Laden der Webseite abbricht, bevor die Bilder vollständig geladen sind, oder die onblur, onfocus und onerror-Ereignisse für das <body>-Tag.



Eine gute Übersicht über die verschiedenen Ereignisse und deren Unterstützung durch Internet Explorer und Netscape Navigator finden Sie auf der Webseite <http://www.teamone.de/selfhtml> unter »JavaScript«, »JavaScript Sprach- elemente«, »Event-

Handler«. Dort finden Sie auch Beispiele zu den einzelnen Ereignissen.

Wenn Sie selbst austesten wollen, ob ein bestimmter Browser ein bestimmtes Ereignis für ein bestimmtes HTML-Element unterstützt, setzen Sie einfach eine Webseite auf, die das betreffende HTML-Element enthält und verbinden Sie das Ereignis mit dem Aufruf eines Meldungsfensters (nicht in Zusammenhang mit onfocus oder onblur).

Listing 10.1: test.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Testseite</title>
  <meta http-equiv="Content-Script-Type" content="text/javascript" />
</head>
<body>
<h1>Testseite für JavaScript-Ereignisse</h1>
<p onclick="alert('Hallo');">Dies ist ein Absatz, für den das onclick-Ereignis
definiert wurde. Klicken Sie auf den Absatz, um die Ereignisverarbeitung zu
testen.</p>
</body>
</html>
```



Abbildung 10.1: onclick-Ereignis für <p>-Tag



Rufen Sie nie im onblur-Ereignis einer Webseite (<body>-Tag) ein anderes Fenster (inklusive Meldungsfenster, Eingabeaufforderung, etc.) auf. Beim Aufrufen des neuen Fensters verliert das ursprüngliche Fenster den Fokus. Kehrt der Webbesucher zu dem ursprünglichen Fenster zurück, erhält dieses wieder den Fokus und das onblur-Ereignis wird wieder ausgeführt (mit dem Resultat, dass das andere Fenster erneut aufgerufen wird). So entsteht eine Endlosschleife, aus der der Websurfer nicht mehr ausbrechen kann.

Das onclick-Ereignis für Bilder

Die folgende Webseite zeigt dem Besucher eine Auswahl von vier Landesflaggen und fordert ihn auf, die Flagge Sri Lankas herauszufinden und anzuklicken.



Abbildung 10.2: Webseite zu Listing 10.2

Der HTML- und JavaScript-Code dieser Webseite sieht wie folgt aus.

Listing 10.2: flaggen1.html - onclick-Ereignis für Bilder (IE und Net 6)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Flaggen</title>
  <meta http-equiv="Content-Script-Type" content="text/javascript" />
</head>
<body style="background-color: #F0F0F0">
<h1>Welche der folgenden Flaggen ist die Flagge von Sri Lanka?</h1>
<table>
  <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
  </tr>
</table>
<br />
```

```
<p>Klicken Sie einfach auf die Flagge.</p>
</body>
</html>
```

Im Internet Explorer 5 oder im Netscape 6-Browser funktioniert diese Webseite ganz wie geplant. Websurfer, die mit dem Navigator unterwegs sind, werden von dieser Webseite aber herbe enttäuscht, denn da der Navigator das onclick-Ereignis nicht für -Tags unterstützt, werden Navigator-Nutzer vergeblich auf die einzelnen Flaggen klicken.

Will man die Navigator-Nutzer nicht ausgrenzen, gibt es zwei Möglichkeiten:

- Sie fordern sie auf, sich den Netscape 6-Browser zu besorgen
- Sie implementieren Ihre Webseite so, dass sie auch im Navigator korrekt funktioniert.

Letzteres geht nur mit einem Trick. Man muss das Bild in ein Anker-Element mit href- Attribut einschließen. Für Anker-Elemente unterstützt nämlich auch der Navigator das onclick-Ereignis.

Listing 10.3: Auszug aus flaggen1b.html - onclick-Ereignis für Bilder (IE, Nav4, Net6)

```
<table>
<tr>
<td><a href="#" onclick="alert('Dies ist die Flagge des Senegal');">
</></td>
<td><a href="#" onclick="alert('Dies ist die Flagge Japans');">
</></td>
<td><a href="#" onclick="alert('Richtig!');">
</></td>
<td><a href="#" onclick="alert('Dies ist die Flagge Finnlands');">
</></td>
</tr>
</table>
```

Beachten Sie, dass das href-Attribut in dem Anker-Element gesetzt sein muss. Damit der Browser aber nicht beim Anklicken des Bildes zu einer anderen Webseite oder Teststelle wechselt, geben wir als Wert zu href eine leere Textmarke "#" an.



Wenn Sie ein Bild in ein Anker-Element einschließen, legen die meisten Browser einen Rahmen um das Bild. Wenn Sie dies für alle gängigen Browser verhindern wollen, setzen Sie das deprecated-Attribut border auf 0: .



In Abschnitt 10.5 werden wir uns ausführlicher mit der Erstellung browserunabhängiger Webseiten befassen.

Argumente an Ereignisbehandlungsfunktionen übergeben

Wie jeder JavaScript-Funktion kann man auch einer Ereignisbehandlungsfunktion ein oder mehrere Argumente übergeben.

Ein speziell für die Ereignisbehandlung recht interessantes Argument ist das Schlüsselwort `this`. Wie Sie aus Kapitel 9.5.2 wissen, repräsentiert das Schlüsselwort `this` in JavaScript-Methoden das jeweils aktuelle Objekt. In der Ereignisbehandlung repräsentiert es das HTML-Element, für das das Ereignis ausgelöst wurde.

Listing 10.4: `flaggen2.html` - `this`-Argument in Ereignisfunktionen

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Flaggen</title>
  <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
  function falsch(elem)
  {
    elem.src = "falsch.gif";
  }
  function richtig(elem)
  {
    elem.src = "richtig.gif";
  }
</script>
</head>
<body style="background-color: #F0F0F0">
<h1>Welche der folgenden Flaggen ist die Flagge von Sri Lanka?</h1>
<table>
  <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
  </tr>
</table>
<br />
<p>Klicken Sie einfach auf die Flagge.</p>
</body>
</html>
```

In diesem Beispiel werden zwei Ereignisbehandlungsfunktionen verwendet - je nachdem, ob die richtige oder die falsche Flagge angeklickt wurde. Beiden Funktionen wird jeweils das auslösende HTML-Element als Argument übergeben (`this`) und beide Funktionen speichern dieses Element in ihrem Parameter `elem`. Da `elem` in den JavaScript-Funktionen ein ``-Element repräsentiert, verfügt `elem` über eine `src`-Eigenschaft. Indem wir dieser Eigenschaft den URL eines anderen Bildes zuweisen, wechseln wir das Bild des ``-Elements.



Abbildung 10.3: Bildwechsel durch Ereignisbehandlung



Beachten Sie, dass dieses Beispiel so nicht im Navigator funktioniert (wohl aber in IE 5 und Netscape 6). Wenn Sie die Bilder in Anker-Elemente fassen und deren onclick-Ereignisse mit den Funktionen falsch() und richtig() verbinden, bezieht sich das this-Argument auf den Anker und nicht auf das darin enthaltene Bild. Man kann also nicht über den Parameter auf das Bild zugreifen, sondern muss einen ganz anderen Weg wählen, beispielsweise: `document.images[0].src = "falsch.gif";`. Mehr hierzu erfahren Sie in Abschnitt 10.4.

10.2 Das event-Objekt

Wenn wir eine Ereignisbehandlungsfunktion aufsetzen, wissen wir für welche Objekte und welche Ereignisse diese Funktion aufgerufen wird (wobei ein und dieselbe Funktion mit mehreren Objekte und gegebenenfalls sogar mit mehreren Ereignisse verbunden werden kann). Doch manchmal reicht diese Information nicht, um den Code der Funktion aufzusetzen. Vielleicht soll die Funktion Mausklicks in Bildern abfangen und, je nachdem in welche Bildbereiche geklickt wurde, unterschiedlich reagieren. Dann benötigen wir Informationen darüber, auf welchen Koordinaten der Mausklick erfolgte. Oder wir wollen unterscheiden, ob der Webbesucher beim Auslösen des Ereignisses die (Shift)-Taste gedrückt hat oder nicht. Zum Abfragen solcher Informationen muss man auf das event-Objekt zugreifen.

Wie man auf das Objekt zugreift, hängt davon ab, in welchem Browser die Webseite angezeigt wird. Zwar kennen sowohl der Internet Explorer als auch die Netscape-Browser ein event-Objekt, doch werden diese auf unterschiedliche Weise zur Verfügung gestellt und weisen auch uneinheitlich benannte Eigenschaften auf.

Wir werden uns im Folgenden zuerst die event-Objekt der beiden Browser-Typen anschauen und dann eine Lösung vorstellen, wie man beiden Browsern gerecht werden kann.

Das event-Objekt im Internet Explorer

Im Internet Explorer wird das event-Objekt als Eigenschaft des globalen window-Objekts zur Verfügung gestellt und ist dabei stets verfügbar.

Tritt ein Ereignis ein, aktualisiert der Browser die Eigenschaften des event-Objekts mit den Daten aus dem aktuellen Ereignis. Wir brauchen diese nur noch abzufragen:

```
1</sup>                                                       |
| layerY      | y-Koordinate von Mausereignissen relativ zum übergeordneten Layer-Element.                                                                   |
| pageX       | x-Koordinate von Mausereignissen relativ zur Webseite.                                                                                       |
| pageY       | y-Koordinate von Mausereignissen relativ zur Webseite.                                                                                       |
| screenX     | x-Koordinate von Mausereignissen relativ zum Bildschirm.                                                                                     |
| screenY     | y-Koordinate von Mausereignissen relativ zum Bildschirm.                                                                                     |
| target      | Das HTML-Element, für das das Ereignis ausgelöst wurde.                                                                                      |
| ctrlKey     | Gibt an, ob die (Strg)-Taste gedrückt war.                                                                                                   |
| type        | Typ des Ereignisses (interessant, wenn man mehrere verschiedene Ereignisse mit einer JavaScript-Funktion verbindet).                         |
| which       | Gibt an, welche Maustaste während des Ereignisses gedrückt war.                                                                              |
| width       | Breite des Fensters oder Frames.                                                                                                             |

**Tabelle 10.3: Die wichtigsten Eigenschaften des event-Objekts der Netscape-Browser**

1

Das LAYER-Element ist eine Erfindung des Netscape Navigators, mit der man Webseiten in Schichten aufbauen konnte (sozusagen als übereinanderliegende transparente Folien). Allerdings wurde es nie in den offiziellen HTML-Standard aufgenommen (dort favorisierte man die Positionierung mit Stylesheets). Im Netscape 6-Browser wird es nicht mehr unterstützt.

## Beispiel

**Listing 10.5: flaggen3.html - Zugriff auf das event-Objekt**

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Flaggen</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
 function falsch(e)
 {
 if (navigator.appName.indexOf("Microsoft") != -1)
 {
 var elem = window.event.srcElement;
 elem.src = "falsch.gif";
 }
 else if (navigator.appName.indexOf("Netscape") != -1
 && parseInt(navigator.appVersion) > 4)
 {
 var elem = e.target;
 elem.src = "falsch.gif";
 }
 }
 function richtig(e)
 {
 if (navigator.appName.indexOf("Microsoft") != -1)
 {
 var elem = window.event.srcElement;
 elem.src = "richtig.gif";
 }
 else if (navigator.appName.indexOf("Netscape") != -1
 && parseInt(navigator.appVersion) > 4)
 {
 var elem = e.target;
 elem.src = "richtig.gif";
 }
 }
</script>
</head>
<body style="background-color: #F0F0F0">
<h1>Welche der folgenden Flaggen ist die Flagge von Sri Lanka?</h1>
<table>
 <tr>
 <td></td>
 <td></td>
 <td></td>
 <td></td>
 </tr>
</table>

<p>Klicken Sie einfach auf die Flagge.</p>
</body>
</html>

```

## 10.3 Erweiterte Möglichkeiten zum Abfangen von Ereignissen

Bisher hatten wir stets eine feste Verbindung zwischen HTML-Element und Ereignis. Daneben gibt es aber auch die Möglichkeit, Ereignisse aus JavaScript-Code heraus auszulösen oder an übergeordnete HTML-Elemente weiterzureichen.

## Ereignisse simulieren

In den meisten Fällen legen Sie als Autor einer Webseite fest, für welche HTML-Elemente welche Ereignisse bearbeitet werden, und der Besucher der Webseite löst die Ereignisse mit seiner Maus oder seiner Tastatur aus.

Es werden sich aber unter Umständen auch einmal Situationen ergeben, in denen Sie ein Ereignis selbst, über Ihren JavaScript-Code, auslösen möchten.

Im Grunde ist dies ganz einfach. Sie müssen lediglich die passende Methode des HTML- Elements aufrufen. Die Methoden zum Auslösen der Ereignisse heißen ebenso wie die Ereignisattribute allerdings ohne das »on«-Präfix. Die einzige Schwierigkeit dabei ist, einen Weg zu finden, wie Sie auf das HTML-Element, dessen Ereignis simuliert werden soll, zugreifen können.

- Ein Weg führt über das document-Objekt und dessen Array-Eigenschaften: `document.anchors[]`, `document.applets[]`, `document.forms[]`, `document.images[]`, `document.links[]`.

Wenn Sie beispielsweise das Klickereignis des Dokuments auslösen wollen, schreiben Sie

```
document.click();
```

Wenn Sie das mouseover-Ereignis des ersten Hyperlinks auf der Webseite auslösen wollen, schreiben Sie

```
document.links[0].mouseover();
```

- Wenn das Element über ein id-Attribut verfügt, kann man es über dieses identifizieren. Hierfür gibt es in den verschiedenen Browsern unterschiedliche Ansätze. Wir werden uns damit in Abschnitt 10.4 befassen.
- Eventuell können Sie über das this-Schlüsselwort (siehe Kapitel 9.5 und Abschnitt 10.1.4) oder das event-Objekt (siehe Abschnitt 10.2) auf das Objekt zugreifen.

## Globale Ereignisbearbeitung im Internet Explorer

HTML-Elemente können nacheinander definiert werden, sie können aber auch ineinander verschachtelt werden (wobei das äußere Element das innere Element stets ganz einschließt). Betrachten wir zum Beispiel folgenden Code:

```
<body>
<p>Ereignisse können - in Abhängigkeit vom Browser - auf verschiedene
Weise in der HTML-Elementhierarchie weitergereicht werden.</p>
</body>
```

Hier ist das innerste Element die fett formatierte Textpassage in den `<b>`-Tags. Um dieses Element liegt das `<p>`-Element, das selbst wieder in das `<body>`-Element eingeschlossen ist.

```

```





Wenn Sie verhindern wollen, dass ein Ereignis an das umliegende HTML-Element weitergereicht wird, setzen Sie in der Ereignisfunktion des letzten HTML-Elements die event-Eigenschaft `cancelBubble` auf `true`: `event.cancelBubble = true`;

## Globale Ereignisbearbeitung im Netscape Navigator

Die Ereignisbearbeitung im Netscape Navigator ist nahezu das glatte Gegenteil zur Ereignisbearbeitung im Internet Explorer.

- Die Ereignisse werden vom übergeordneten HTML-Element an die eingeschlossenen Elemente weitergereicht.
- Die Ereignisse müssen zum Weiterreichen explizit erneut ausgelöst werden (entweder durch Simulation des Ereignisses (siehe oben) oder durch Aufruf der event-Methode `routeHandler`: `e.routeHandler()`).
- Viele Ereignisse werden gar nicht unterstützt (beispielsweise `onclick`, `onmouse...` für `<img>`)

Hinzukommen noch einige weitere Besonderheiten.

An der Spitze der Ereignisbearbeitung stehen das `window`- (Browserfenster), das `document`- (Webseite in Browserfenster) und das `layer`-Objekt (Layer-Ebene in Webseite).

Um für diese Objekte Ereignisse abzufangen, muss man einen besonderen Weg einschlagen:

1. Man weist die Ereignisbehandlungsfunktion der Ereignis-Eigenschaft der Objekte zu.

```
window.onclick = fkt_w;
document.onclick = fkt_d;
```

Die Ereignis-Eigenschaften heißen in diesem Falle genauso wie die Ereignis-Attribute der entsprechenden HTML-Elemente.

In gleicher Weise kann man übrigens für den Navigator auch Ereignisfunktionen zu beliebigen anderen Objekten zuweisen (beispielsweise `document.links[0].onmousemove = meinefkt`).

2. Man muss den Browser explizit anweisen, die betreffenden Ereignisse für das `window`- bzw. `document`-Objekt abzufangen.

Hierzu verwendet man die Methode `captureEvents()` und übergibt ihr den Typ der abzufangenden Eigenschaften. Die Ereignistypen lauten wie die Ereignisse (ohne on-Präfix) und werden in Großbuchstaben geschrieben.

```
window.captureEvents(Event.CLICK);
document.captureEvents(Event.CLICK);
```

Die Webseite *globEvent\_Nav.html* verdeutlicht, wie man im Netscape Navigator Ereignisse global abfangen kann. Sie können die Webseite auch von der Buch-CD laden und ein wenig mit der Ereignisbearbeitung im Navigator herumexperimentieren.

## Listing 10.7: globEvent\_Nav.html - Ereignisverarbeitung im Netscape Navigator

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Ereignisse</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
 function init()
 {
 window.captureEvents(Event.CLICK);
 document.captureEvents(Event.CLICK);
 window.onclick = fkt_w;
 document.onclick = fkt_d;
 }
 function fkt_w(e)
 {
 alert("Grüsse vom Browserfenster!");
 e.routeHandler(); // Ereignis weiterleiten
 }
 function fkt_d(e)
 {
 alert("Grüsse von der Webseite!");
 e.routeHandler(); // Ereignis weiterleiten
 }
 function fkt_a()
 {
 alert("Grüsse vom Link!");
 }
</script>
</head>
<body onload="init()">

</body>
</html>
```

## Globale Ereignisbearbeitung im Netscape 6-Browser

Der Netscape 6-Browser verwendet eine Kombination aus den Ereignisbearbeitungsmodellen des Internet Explorers und des Netscape 6-Browsers - und zwar auf zwei verschiedene Weisen.

### Unterstützung der Ereignismodelle der anderen Browser

Der Netscape 6-Browser kann sowohl den IE-kompatiblen Code zum Abfangen der aufsteigenden Ereignisse als auch den Navigator-kompatiblen Code verarbeiten.



*Der Netscape 6-Browser verwendet als Standardmodell der Ereignisbearbeitung das Bubbling-Modell. das heißt, die Ereignisse werden von den eingeschlossenen an die umliegenden HTML-Elemente weitergereicht. Dies gilt auch für Navigator-kompatiblen Code, dessen Ereignisse im Navigator in umgekehrter Richtung - von den umliegenden zu den eingeschlossenen Elementen - weitergereicht würden.*

## Eigenes Ereignismodell mit Abfangen von Ereignissen beim Ab- und Aufsteigen

Darüber hinaus definiert der Netscape-Browser ein eigenes Modell, das zwei Erweiterungen bietet:

- Man kann Ereignisse in beiden Richtungen abfangen.
- Man kann zu einem Ereignis mehrere Ereignisbehandlungsfunktionen einrichten.

Dieses Modell basiert darauf, dass Sie die Ereignisfunktionen mit Hilfe der HTML- Objektmethode `addEventListener()` einrichten.

1. Den Code zur Verknüpfung der Ereignisfunktionen mit den Ereignissen setzt man am besten in einer eigenen Funktion auf, die beim Laden der Webseite ausgeführt wird.

```
<head>
...
<script type="text/javascript">
 function init()
 {
 ...
 }
</script>
</head>
<body onload="init()">
```

2. Dann beschaffen Sie sich ein Objekt, das das HTML-Element repräsentiert, dessen Ereignis Sie abfangen wollen.

Am einfachsten weist man dazu dem HTML-Element ein eindeutiges `id`-Attribut zu

```
<body id="body" onload="init()">
```

und lässt sich das Objekt von der `document`-Methode `getElementById()` zurückliefern:

```
function init()
{
 obj = document.getElementById("body");
```

3. Für dieses Objekt ruft man dann die Methode `addEventListener()` auf und übergibt dieser

- den Typ des zu bearbeitenden Ereignisses (in Kleinbuchstaben)
- den Namen der Ereignisbehandlungsfunktion
- einen Booleschen Wert, der anzeigt, ob das Ereignis beim Ab- oder beim Aufsteigen abgefangen werden soll.

```
function init()
{
 obj = document.getElementById("body");
 obj.addEventListener("click", fkt_w, false);
```

}



*Wenn Sie die Ereignisse wie im Internet Explorer aufsteigend (von innen nach außen) abfangen wollen, übergeben Sie als Argument für den letzten Parameter false. Wenn Sie mit true Ereignisse beim Absteigen abfangen wollen, müssen Sie beachten, dass dies nicht mit allen Ereignissen geht, sondern nur mit den Ereignissen, die auch der Navigator unterstützt.*

In Listing 10.8 sehen Sie die Adaption der Webseite *globEvent\_IE.html* (Listing 10.6) für das Ereignismodell des Netscape-Browsers. (Beachten Sie, dass Sie die Webseite aus Listing 10.6 aber auch unverändert im Netscape-Browser ausführen können.) Listing 10.9 ist eine Adaption der Webseite *globEvent\_Nav.html* (Listing 10.7) und demonstriert auch das Einrichten mehrerer Ereignisfunktionen für ein Ereignis sowie das Abfangen beim Ab- und Aufsteigen.

### Listing 10.8: globEvent\_Net6a.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Ereignisse</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
 function init()
 {
 obj = document.getElementById("body");
 obj.addEventListener("click", fkt_w, false);
 obj = document.getElementById("absatz");
 obj.addEventListener("click", fkt_p, false);
 obj = document.getElementById("fett");
 obj.addEventListener("click", fkt_b, false);
 }
 function fkt_w()
 {
 alert("Grüsse von der Webseite!");
 }
 function fkt_p()
 {
 alert("Grüsse vom Absatz!");
 }
 function fkt_b()
 {
 alert("Grüsse vom fetten Einschub!");
 }
</script>
</head>
<body id="body" onload="init()">
<p id="absatz">Ereignisse können - <b id="fett">in Abhängigkeit vom Browser
- auf verschiedene Weise in der HTML-Elementhierarchie weitergereicht
```

```
werden.</p>
</body>
</html>
```

### Listing 10.9: globEvent\_Net6b.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Ereignisse</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
 function init()
 {
 obj = document.getElementById("body");
 obj.addEventListener("click", fkt_w, true);
 obj.addEventListener("click", fkt_w, false);
 obj = document.getElementById("a");
 obj.addEventListener("click", fkt_a, true);
 obj.addEventListener("click", fkt_a, false);
 }
 function fkt_w(e)
 {
 alert("Grüsse von der Webseite!");
 }
 function fkt_a()
 {
 alert("Grüsse vom Link!");
 }
</script>
</head>
<body onload="init()" id="body">

</body>
</html>
```

## Allgemeine globale Ereignisbearbeitung

Wie soll man angesichts dieses Wirrwarrs an browserspezifischen Ereignismodellen seine Webseiten aufsetzen?

Nun, so schlimm ist es gar nicht. Betrachtet man die Modelle, kann man feststellen,

- dass der Netscape 6-Browser und der Internet Explorer sowohl in der Standardereignisbearbeitung (Verwendung der HTML-Ereignisattribute) als auch der globalen Ereignisbearbeitung praktisch vollständig kompatibel sind, sofern man auf das Netscape 6-spezifische Ereignismodell mit seinen erweiterten Möglichkeiten verzichtet.
- dass der Navigator in seiner Ereignisbearbeitung sehr eingeschränkt ist und viele Ereignisse gar nicht unterstützt.

Ausgehend von diesen Prämissen, kann man zwei Wege zur Planung einer Webseite einschlagen.

## Sie wollen unbedingt auch den Navigator unterstützen

In diesem Falle sollten Sie auf alle Ereignisse verzichten, die der Navigator nicht unterstützt. Klicks auf Bilder können Sie simulieren, indem Sie die Bilder in Anker-Elemente einschließen (siehe Listing 10.3).

Wenn Sie von der globalen Ereignisbearbeitung Gebrauch machen wollen, müssen Sie separaten Code für den Navigator und den Internet Explorer aufsetzen (siehe Abschnitt 10.5).

## Sie wollen den Navigator nicht unbedingt unterstützen

Setzen Sie Ihre Webseite so auf, dass sie im Internet Explorer korrekt funktioniert. Testen Sie die Webseite dann im Netscape-6-Browser. Bezüglich der Ereignisbearbeitung sollte es dabei keine Schwierigkeiten geben.



*Eventuell enthalten Ihre Ereignisfunktionen jedoch Anweisungen, die nur vom Internet Explorer verstanden werden (beispielsweise wenn Sie über `document.all` auf einzelne HTML-Elemente zugreifen, siehe nachfolgender Abschnitt). Dann müssen Sie in den Ereignisfunktionen separaten Code für den Internet Explorer und den Netscape 6-Browser aufsetzen.*

Testen Sie die Webseite im Navigator. Richten Sie für den Navigator alternativen Code ein, damit die Webseite im Navigator einigermaßen adäquat angezeigt wird, oder verweisen Sie darauf, dass die Leser den Netscape 6-Browser herunterladen sollten (siehe Abschnitt 10.5).

## 10.4 Das DOM-Modell

JavaScript wäre als Websprache bei weitem nicht so interessant und weit verbreitet, wenn es keine Möglichkeit gäbe, mit JavaScript auf die verschiedenen HTML-Elemente einer Webseite zuzugreifen. Viele interessante Einsatzgebiete für JavaScript ergeben sich erst daraus, dass man mit JavaScript in eine Webseite schreiben kann, dass man die Werte von Eingabe- oder Optionsfeldern abfragen kann oder dass man ein Bild wechselt oder verschiebt. Wie dies geht und welche Möglichkeiten es hierfür gibt, wird aber weniger von der JavaScript-Sprachspezifikation als vielmehr von den Browsern und der DOM<sup>1</sup>-Spezifikation festgelegt.

### Die Browser und die DOM-Spezifikation

Wie bereits erwähnt, ging die Initiative bei der Entwicklung von JavaScript bislang vornehmlich von den Browsern aus. Diese definierten auch das `document`-Objekt, über das wir auf die Eigenschaften und untergeordneten HTML-Elemente einer Webseite zugreifen können.

Schauen wir uns ein wenig genauer an, wie man über das `document`-Objekt auf die HTML-Elemente einer Webseite zugreifen kann.

Webseiten bestehen aus Absätzen, Überschriften, Bildern, Anker und Hyperlinks, Formularen, Tabellen, etc. Unter diesen verschiedenen HTML-Elementtypen erachtete man die Anker, Applets, Formulare, Bilder und Hyperlinks als besonders interessant für den Zugriff aus JavaScript-Code. Daher boten die großen Browser bald Array-Eigenschaften an, über die man auf diese Elemente zugreifen konnte:

```
document.anchors[]
document.applets[]
document.forms[]
document.images[]
document.links[]
```

Bis dahin sah die Entwicklung im Wesentlichen so aus, dass Netscape sein JavaScript- DHTML-Modell ständig ausbaute und mit neuen Ideen immer weiter vorantrieb, während Microsoft atemlos hinterher rannte und bemüht war, die neuen Konzepte auch in seinem Internet Explorer zu unterstützen.

Zwischen Ende 96 und Anfang 98 spaltete sich die Entwicklung aber plötzlich in drei Äste auf - mit fatalen Folgen für den Netscape Navigator.

## Das <layer>-Tag des Navigators 4

Netscape entwickelte das Layer-Konzept zum schichtweisen Aufbau von Webseiten. Dieses Konzept brachte zwei Neuerungen:

- Man konnte Webseiten schichtweise aufbauen. Man kann sich diese Schichten wie Transparenzfolien vorstellen. Auf jede Folie »malt« man ein, zwei oder mehrere HTML-Elemente. Zum Schluss legt man die Folien übereinander und erhält die fertige Webseite.
- Des Weiteren führte Netscape mit den Layer-Schichten die freie Positionierung von HTML-Elementen ein.



*Frei positionierte Elemente (Stileigenschaft `position: absolute`), die über eine `id`-Kennung verfügen, können im Navigator auch über ihren `id`-Wert angesprochen werden: `document.idWert.src = ...` Wir werden darauf noch einmal im Kapitel 16.3 eingehen, ansonsten werden wir uns mit dem `<layer>`-Tag und den daran geknüpften Techniken nicht weiter befassen, da diese mit zunehmender Verbreitung des Internet Explorers und des Netscape 6-Browser wohl schon bald der Vergangenheit angehören werden.*

## Das all-Objekt des Internet Explorers 4

Der Internet Explorer führte das `document.all`-Objekt ein. Über dieses Objekt konnte (und kann) man auf alle (!) HTML-Elemente einer Webseite zugreifen. Der Zugriff erfolgt entweder über die Eigenschaft `tags`, Angabe des Tag-Typs (beispielsweise "h1") und einen Index:

```
document.all.tags("h1")[0].style.fontSize = "60px";
```

oder über den `id`-Wert des Elements:

```
document.all.idWert.style.fontSize = "60px";
```

In letzterem Fall kann man sogar auf das Präfix `document.all` verzichten:

```
idWert.style.fontSize = "60px";
```



*Bei allem Komfort, den das all-Objekt bietet, werden wir nicht näher auf es eingehen, da es nicht W3C-konform ist und der Internet Explorer seit der Version 5 als Alternative auch die offiziellen DOM-Methoden, beispielsweise `document.getElementById()`, unterstützt (siehe nachfolgenden Abschnitte).*

## ECMA, CSS und DOM

Im gleichen Zeitraum wurde eine Reihe von HTML- und JavaScript-relevanten Webstandards ausgearbeitet und vorgestellt.

- Der ECMA-Standard legte - auf der Basis der im Navigator und Internet Explorer verwendeten Interpreter - eine allgemein verbindliche Syntax für JavaScript fest.
- Die CSS-Spezifikation legte die Regeln für die Formatierung mit Stylesheets fest. Die CSS2-Spezifikation, die im Mai 1998 veröffentlicht wurde, regelte zudem die freie Positionierung von HTML-Elementen via Stylesheets (siehe Kapitel 5). Durch die offizielle Aufnahme und Unterstützung der CSS-Stylesheets seitens des HTML- Standards geriet das Layer-Konzept von Netscape ins Abseits.
- Im Oktober 1998 wurde die W3C-Empfehlung für das Document Object Model (DOM) vorgestellt. Diese Empfehlung regelt wie die Elemente einer XML/HTML- Seite in einer hierarchisch angeordneten Struktur repräsentiert und von Websprachen wie Java oder JavaScript aus bearbeitet werden können.

Plötzlich war der Internet Explorer ein gutes Stück näher an den offiziellen Webstandards als der Netscape Navigator. Der Internet Explorer 4/5 unterstützte - wie vom HTML-4- Standard gefordert - weit mehr Ereignisse als der in dieser Beziehung sehr restriktive Navigator 4 (obwohl auch in diesem neue Ereignisse hinzugekommen waren). Er setzte weite Teile der CSS1- und CSS2-Spezifikation um und machte die Stilinformationen für JavaScript-Code zugänglich, während der Navigator auch hier nur zögerlich reagierte und die freie Positionierung mit seinem Layer-Konzept verwob, das aber - wie man im Nachhinein feststellen kann - mit der offiziellen Bevorzugung des CSS2-Standards zum Scheitern verurteilt war. Schließlich lag der Internet Explorer auch näher am DOM- Standard, denn er implementierte neben seinem nicht DOM-kompatiblen all-Objekt auch schon die ersten DOM-Methoden.

## Der Netscape 6-Browser

Die folgenden Jahre wurden zu einem Machtkampf zwischen dem Internet Explorer und dem Navigator, den der Internet Explorer mittlerweile klar gewonnen hat. Er bot mehr Funktionalität und war näher an den offiziellen Standards<sup>2</sup>.

Nun hat Netscape mit seinem neuen Netscape 6-Browser dem alten Layer-Konzept und der Politik der zögerlichen Unterstützung der offiziellen Standards den Rücken gekehrt. Vom Saulus zum Paulus gewandelt, bemüht sich der Netscape-Browser jetzt um vorbildliche Umsetzung der offiziellen Standards. Eine neue Runde ist eingeläutet, wir dürfen gespannt sein, wie es weiter geht.

## Das document-Objekt

Das document-Objekt wird vom Browser erzeugt und repräsentiert die aktuell im Browserfenster angezeigte Webseite. Es enthält eine Reihe von interessanten Eigenschaften und Methoden.

## Die Eigenschaften

Eigenschaft	Beschreibung
alinkColor	Farbe aktivierter Links
anchors[]	Array, über das man auf die Anker-Elemente der Webseite zugreifen kann
applets[]	Array, über das man auf die Applet-Elemente der Webseite zugreifen kann
body	Das <body>-Element der Webseite
bgColor	Hintergrundfarbe
cookie	Von der Webseite gesetzte Cookies
domain	Domänenname des Webservers, von dem die Webseite stammt
fgColor	Vordergrundfarbe
forms[]	Array, über das man auf die Formulare der Webseite zugreifen kann
images[]	Array, über das man auf die <img>-Elemente der Webseite zugreifen kann
linkColor	Farbe von Hyperlinks
links[]	Array, über das man auf die Hyperlinks der Webseite zugreifen kann
referrer	URL, von dem der Besucher zu dieser Seite gekommen ist
title	Titel der Webseite
URL	URL der Webseite
vlinkColor	Farbe besuchter Hyperlinks

**Tabelle 10.4: document-Eigenschaften**

Je nach Browser verfügt das document-Objekt noch über einige weitere Eigenschaften, aber die oben aufgeführten haben Eingang in den offiziellen DOM-Standard gefunden und werden weitestgehend unterstützt.



*Die Attribute `alinkColor`, `bgColor`, `fgColor`, `linkColor` und `vlinkColor` sind in der DOM-Spezifikation allerdings nicht als Eigenschaften des document-Objekts, sondern als Eigenschaften des body-Objekts definiert. (IE 5 und Netscape 6 unterstützen den Zugriff über die body-Eigenschaft.)*

Der Zugriff auf die Eigenschaften des document-Objekts ist die Einfachheit selbst:

```
var str = document.title; // Wert der Eigenschaft abfragen
document.bgColor = "green"; // Wert der Eigenschaft neu setzen
```

Während es jedoch immer möglich ist, den Wert einer Eigenschaft abzufragen, kann man nicht jede Eigenschaft beliebig verändern. Beispielsweise können Sie zwar die Hintergrundfarbe einer geladenen Webseite ändern, nicht aber die Vordergrundfarbe oder den Titel.

Interessant sind auch die Array-Eigenschaften des document-Objekts. Während der Browser eine Webseite lädt und aufbaut, registriert er alle in der Webseite definierten Anker, Applets, Formulare, Bilder und Hyperlinks. Für jedes Vorkommen dieser HTML-Elemente erzeugt er ein eigenes Objekt und trägt es in das zugehörige Array ein. Nach dem Laden der Webseite können wir in unserem JavaScript-Code auf die betreffenden HTML-Elemente über die Arrays zugreifen.

```
str = document.links[0].href; // liefert den URL des ersten Links der
 // Webseite zurück
document.images[3].src = "bld_a.gif" // weist dem 4. -Tag ein
 // neues Bild zu
```

Diese Eigenschaften wurden übrigens nur deshalb in den DOM-Standard aufgenommen, weil sie zu der Zeit, da der DOM-Standard erarbeitet wurde, bereits seit längerem von den wichtigen Browsern unterstützt und von Webautoren weidlich genutzt wurden. Um nicht zu sehr gegen den Strom zu schwimmen, legalisierte man den vorhandenen Code nachträglich und nahm die Eigenschaften in den Standard auf, obwohl der DOM-Standard an sich alternative und leistungsfähigere Möglichkeiten zum Zugriff auf die einzelnen HTML-Elemente vorsieht.

## Die Methoden

Methode	Beschreibung
close()	Dokument abschließen
open()	Dokument öffnen
write()	In Dokument schreiben
writeln()	In Dokument schreiben (inklusive abschließendem Zeilenumbruch)
getElementById()	HTML-Element der angegebenen ID zurückliefern
getElementsByName()	HTML-Element des angegebenen Namens zurückliefern

**Tabelle 10.5: document-Methoden**

Die ersten vier Methoden kennen Sie bereits aus Kapitel 8. Neu sind dagegen die Methoden mit den viel versprechenden Namen getElementById() und getElementsByName(). Diese Methoden sind aber nicht nur für uns neu, sie sind auch insofern neu, als sie nicht von den Browser-Herstellern, sondern vom DOM 1-Standard definiert wurden. Daher werden Sie derzeit auch nur in den neuesten Browserversionen (Internet Explorer 5 und Netscape 6-Browser) unterstützt.

Mit document.getElementById() können Sie auf ein beliebiges HTML-Element zugreifen. Voraussetzung ist allerdings, dass das Element über eine eindeutige id-Kennung verfügt.

Nehmen wir beispielsweise an, Sie hätten eine Webseite mit einem <img>-Tag, dem Sie die id-Kennung "meinBild" zugewiesen haben.

```
<body>
...

...
</body>
```

Dann können Sie sich wie folgt ein JavaScript-Objekt zurückliefern, das das <img>-Element repräsentiert:

```
var obj = document.getElementById("meinBild");
```

Danach kann man wie gewohnt auf die Eigenschaften des Objekts (entsprechen den HTML-Attributen des HTML-Elements) zugreifen.

```
alert(obj.src);
```



*id-Werte sollten immer eindeutig sein, das heißt, es sollte in einer Webseite keine zwei HTML-Elemente mit gleichen id-Kennungen geben. Ansonsten liefert die Methode getElementById() unvorhersehbare Ergebnisse.*

Wenn Sie für das <img>-Element auch ein name-Attribut spezifiziert haben:

```
<body>
...

...
</body>
```

können Sie sich auch mit Hilfe der Methode getElementByName() ein JavaScript-Objekt für das HTML-Element zurückliefern lassen:

```
var obj = document.getElementsByName("bild1");
```

Beachten Sie aber, dass die Methode getElementByName() ein Array aller HTML-Elemente mit dem übergebenen Namen zurückliefert. Wir müssen das gewünschte Objekt also über einen Index ansprechen:

```
alert(obj[0].src);
```



*Beachten Sie, dass die Verwendung des name-Attributs nicht für alle HTML-Elemente gestattet ist und auch dort wo sie erlaubt ist, meist nur noch der Abwärtskompatibilität dient. Verwenden Sie daher bevorzugt das id-Attribut und die Methode getElementById().*

## Das DOM-Modell

Die DOM-Spezifikation des W3C-Konsortiums regelt wie die Elemente einer XML/HTML-Seite in einer hierarchisch angeordneten Struktur repräsentiert und von Websprachen wie Java oder JavaScript aus bearbeitet werden können. Die DOM-Spezifikation ist bei weitem nicht auf das document-Objekt beschränkt, das wir im vorangehenden Abschnitt näher kennen gelernt haben, und es hat ziemlich wenig

mit den traditionell von den Browsern unterstützten Array-Eigenschaften `document.links[]`, `document.images[]`, etc. zu tun.

Das DOM-Modell<sup>3</sup> ist viel weiter gefasst als die traditionellen, schrittweise gewachsenen DHTML-Möglichkeiten der einzelnen Browser. Ziel der DOM-Spezifikation ist es, eine Schnittstelle zu schaffen, die es Programmierern erlaubt XML- und HTML-Dokumente dynamisch zu erstellen, ihre Struktur zu durchforsten und Elemente zu verändern, zu löschen oder hinzuzufügen. Bis auf wenige Ausnahmen soll alles, was in einem XML- oder HTML-Dokument zu finden ist, über die DOM-Schnittstelle verfügbar sein.

## Die Dokumenthierarchie

Das DOM-Spezifikation beruht darauf, dass man jedes Dokument als eine hierarchische Struktur von Knoten interpretiert. Die einzelnen Knoten sind dabei die HTML- und Textelemente des Dokuments. Ganz oben in der Hierarchie steht der Knoten, der das `<body>`-Element repräsentiert. Unter dem `<body>`-Knoten folgen seine Kindknoten. Dies sind alle HTML-Elemente und jegliche Textelemente (die nicht in Tags eingeschlossen ist), die direkt unter dem `<body>`-Element definiert sind.

Allgemein gilt: Wenn ein HTML-Element andere HTML-Elemente einschließt, werden diese als Kindknoten des übergeordneten HTML-Elements interpretiert. Textelemente und HTML-Elemente ohne Inhalt (beispielsweise `<img />` oder `<hr />` können keine Kindknoten haben.

Gemäß diesen Regeln ist es keine große Schwierigkeit die DOM-Präsentation einer Webseite zu finden. Tatsächlich spiegelt diese einfach die Verschachtelung der HTML- Elemente im Code der Webseite wider.

Abbildung 10.4 zeigt das DOM-Modell der Webseite *dom.html* aus Listing 10.10.

### Listing 10.10: dom.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Das DOM-Modell</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
</head>
<body>
<h1>Das DOM-Modell</h1>
<p id="absatz1">Dies ist ein Hyperlink in einem
Textabsatz.</p>
<p>Dies ist ein zweiter Textabsatz.</p>
</body>
</html>
```

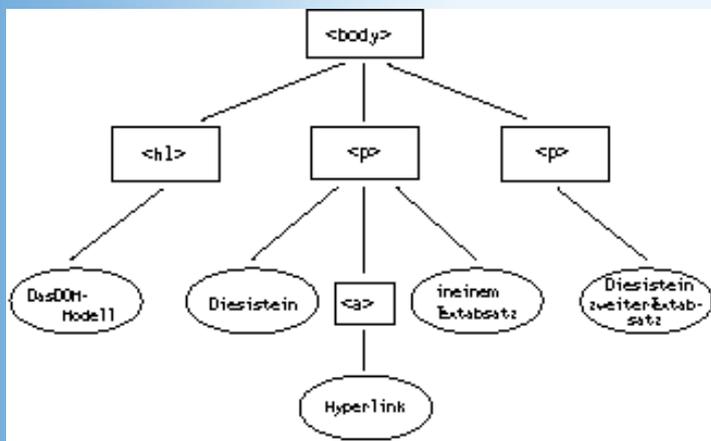


Abbildung 10.4: DOM-Modell zur Webseite aus dom.html

## Das Navigieren in der Dokumenthierarchie

Wie bekommt man Zugriff auf ein bestimmtes HTML-Element einer Webseite?

Hierfür gibt es zwei Möglichkeiten:

- Das Element verfügt über ein id-Attribut und sie lassen es sich direkt durch Aufruf der Methode `document.getElementById()` zurückliefern.
- Sie verwenden die in der DOM-Spezifikation definierten Navigationseigenschaften.

Jeder Knoten verfügt nämlich über Eigenschaften, mit denen man auf seine Kinder, seine Geschwister (Knoten, die das gleiche übergeordnete Elter-Tag haben) oder sein Elter-Tag zugreifen kann.

Eigenschaft	Beschreibung
<code>firstChild</code>	Liefert den ersten Kindknoten des aktuellen Knoten. Liefert null, wenn es keinen Kindknoten gibt.
<code>lastChild</code>	Liefert den letzten Kindknoten des aktuellen Knoten. Liefert null, wenn es keinen Kindknoten gibt.
<code>childNodes</code>	Liefert eine Liste der Kindknoten des aktuellen Knoten.  Die einzelnen Knoten in der Liste können über einen 0-basierten Index angesprochen werden. Die Anzahl der Knoten in der Liste kann über die Eigenschaft <code>length</code> der zurückgelieferten Liste abgefragt werden.
<code>previousSibling</code>	Liefert den vorangehenden Geschwisterknoten des aktuellen Knoten. Liefert null, wenn es keinen solchen Knoten gibt.
<code>nextSibling</code>	Liefert den nachfolgenden Geschwisterknoten des aktuellen Knoten. Liefert null, wenn es keinen solchen Knoten gibt.
<code>parentNode</code>	Liefert den übergeordneten (Elter)-Knoten des aktuellen Knoten. Liefert null, wenn es keinen solchen Knoten gibt (etwa weil dieser Knoten dynamisch entfernt wurde).
<code>ownerDocument</code>	Liefert das <code>document</code> -Objekt des Knotens.

Tabelle 10.6: Knoten-Eigenschaften für die Navigation in der Dokumentstruktur

In Listing 10.11 verwenden wir den direkten Zugriff über die `id`, um die Textfarbe für den ersten `<p>`-Absatz

auf Rot zu setzen.

Von dem Objekt, das den Knoten des ersten <p>-Absatzes repräsentiert, springen wir dann mit Hilfe der Knoteneigenschaften zu dem Hyperlink (erster Kindknoten des <p>-Absatzes) und dem zweiten Absatz (nachfolgendes Geschwister des ersten <p>-Absatzes).

### Listing 10.11: dom.html - erweitert um Skriptcode

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Das DOM-Modell</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
</head>
<body>
<h1>Das DOM-Modell</h1>
<p id="absatz1">Dies ist ein Hyperlink in einem
Textabsatz.</p>
<p>Dies ist ein zweiter Textabsatz.</p>
<script type="text/javascript">
var pl = document.getElementById("absatz1");
 pl.style.color = "red";
var plKinder = pl.childNodes;
 plKinder[1].style.backgroundColor = "red";
var ueber1 = plKinder[1].parentNode.previousSibling;
 ueber1.style.color = "green";
</script>
</body>
</html>
```

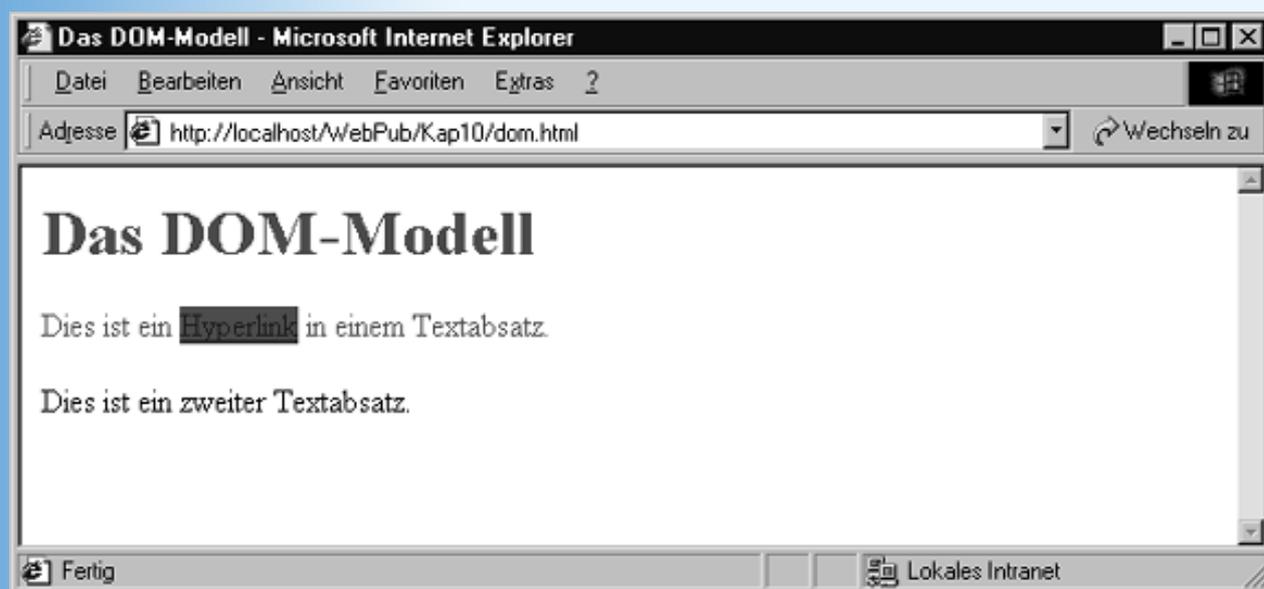


Abbildung 10.5: Sorry - die Farben sind leider verloren gegangen

## Knoten/HTML-Elemente bearbeiten

Angenommen wir haben uns ein Objekt zu einem Knoten in der Dokumentstruktur zurückliefern lassen. Was können wir dann mit diesem Objekt anfangen?

Nun, das hängt zuerst einmal davon ab, um was für einen Knoten es sich handelt.

## Text von Textelementen verändern

Handelt es sich um ein Textelement, können Sie über die Eigenschaft `nodeValue` einen neuen Text zuweisen.

Um beispielsweise den Text des zweiten Absatzes vom *dom.html* zu verändern, könnten Sie schreiben:

```
var p1 = document.getElementById("absatz1");
var p2 = p1.nextSibling;
p2.firstChild.nodeValue = "Dies ist ein neuer Text.";
```

### innerHTML

Der Internet Explorer kennt noch eine weitere Möglichkeit, den Text in einem HTML-Element zu verändern. Voraussetzung ist, dass Sie die `id`-Kennung eines HTML-Elements wissen oder anderweitig Zugriff auf das Element haben (im nachfolgenden Beispiel übergeben wir das Objekt in Form des Schlüsselworts an die Funktion). Ist das Objekt verfügbar, können Sie dem Element über die Eigenschaft `innerHTML` einen neuen Text zuweisen.

```
<script type="text/javascript">
function aendern(elem)
{
 elem.innerHTML = "Dies ist ein neuer Textabsatz."
}
</script>
...
<p onclick="aendern(this)">Dies ist ein Textabsatz.</p>
```

Die Eigenschaft ist etwas einfacher zu verwenden, als der Zugriff über die DOM-Eigenschaft `nodeValue`, da man nicht bis auf die Ebene des Textelements hinabsteigen muss, sondern direkt das HTML-Element manipuliert. Beachten Sie aber, dass die Eigenschaft `innerHTML` proprietär ist, sie wird derzeit sowohl vom Internet Explorer als auch vom Netscape 6-Browser unterstützt.

## HTML-Attribute verändern

Handelt es sich um ein HTML-Element, können Sie sich mit `getAttribute(name)` den Wert eines Attributs zurückliefern lassen, mit `setAttribute(name, wert)` den Wert eines Attributs verändern oder ein neues Attribut zuweisen, mit `removeAttribute(name)` ein Attribut löschen.

Um beispielsweise den Hyperlink aus *dom.html* auf *seite2.html* umzulenken, könnten Sie schreiben:

```
var p1 = document.getElementById("absatz1");
var p1Kinder = p1.childNodes;
p1Kinder[1].setAttribute("href", "seite2.html");
```

## Stileigenschaften verändern

Wenn Sie die Stileigenschaften eines HTML-Elements verändern wollen, gehen Sie nicht über die Methode `setAttribute()`, sondern verwenden die `style`-Eigenschaft des Elements:

```
var p1 = document.getElementById("absatz1");
```

```
p1.style.color = "red";
p1.style.fontSize = "20px";
```

Hierbei ist die Schreibweise der Stileigenschaften zu beachten:

- Bindestriche sind nicht erlaubt, sie fallen weg.
- In zusammengesetzten Wörtern wird jedes Wort außer dem ersten mit einem Großbuchstaben begonnen.

So wird aus font-size die Eigenschaft `fontSize` oder aus background-color die Eigenschaft `backgroundColor`.



*Die obigen Regeln zur Schreibweise von Stileigenschaften gelten in der DOM-Spezifikation übrigens ganz allgemein. Wenn Sie sich obige Beispiele anschauen, werden Sie leicht feststellen, dass die DOM-Methoden nach den gleichen Regeln benannt sind.*

## Noch einmal - das Flaggenbeispiel

Zum Abschluss dieses Themas wollen wir noch einmal das Flaggen-Beispiel vom Anfang dieses Tages aufgreifen und mit Hilfe der DOM-Methoden so implementieren, dass es im Internet Explorer 4/5 und im Netscape 6-Browser korrekt angezeigt wird.

### Listing 10.12: flaggen1.html - onclick-Ereignis für Bilder (IE und Net 6)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Flaggen</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
 function loesen(str)
 {
 if (str == "bld3")
 document.getElementById(str).src = "richtig.gif";
 else
 document.getElementById(str).src = "falsch.gif";
 }
</script>
</head>
<body style="background-color: #F0F0F0">
<h1>Welche der folgenden Flaggen ist die Flagge von Sri Lanka?</h1>
<table>
 <tr>
 <td></td>
 <td></td>
 <td></td>
 <td></td>
```

```
</tr>
</table>

<p>Klicken Sie einfach auf die Flagge.</p>
</body>
</html>
```

In dieser Version haben wir nur noch eine Funktion zum Bilderwechseln. Das bedeutet, dass wir in der Funktion entscheiden müssen, ob das Bild *richtig.gif* oder das Bild *falsch.gif* einzublenden ist. Und wir müssen natürlich auch sicherstellen, dass die Funktion das Bild wechselt, das angeklickt wurde. Dazu versehen wir jedes `<img>`-Tag mit einer id-Kennung und übergeben den Wert dieser Kennung als Argument an die JavaScript-Funktion.

```

```

In der Funktion können wir anhand des übergebenen id-Wert feststellen, ob die richtige oder eine falsche Flagge angeklickt wurde:

```
function loesen(str)
{
 if (str == "bld3")
 // richtige Flagge wurde angeklickt
 else
 // falsche Flagge wurde angeklickt
}
```

Und wir können den id-Wert nutzen, um mittels `getElementById()` auf das zu wechselnde `<img>`-Tag zuzugreifen.

```
function loesen(str)
{
 if (str == "bld3")
 document.getElementById(str).src = "richtig.gif";
 else
 document.getElementById(str).src = "falsch.gif";
}
```



Abbildung 10.6: Dynamische Bilder mit DOM

## 10.5 Browserunabhängige Lösungen

In diesem Kapitel haben Sie mehr als einmal Hinweise der Art »Dies geht nur in diesem oder jenem Browser«, »Diese Technik wird nur von diesem oder jenem Browser unterstützt, in den anderen Browser müssen Sie ...« gehört. Dabei ging es stets um Techniken, die entweder nur von einigen wenigen Browsern unterstützt werden oder die in verschiedenen Browsern unterschiedlich realisiert werden müssen. Wer solche Techniken nutzen, gleichzeitig aber möglichst alle gängigen Browser unterstützen möchte, der muss nach Tricks suchen, wie er den Code so formulieren kann, dass er in allen Browsern korrekt interpretiert wird.

Manchmal ist dies bereits ohne große Verbiegungen möglich.

Beispielsweise sind sich die aktuellen Versionen der großen Browser doch sehr ähnlich und sofern Sie für den Zugriff auf die HTML-Elemente die Methode `document.getElementById()` statt dem Unterobjekt `document.all` verwenden, sollte es keine allzu große Schwierigkeit sein, Code aufzusetzen, der im Internet Explorer 5 wie im Netscape 6-Browser einwandfrei läuft (allerdings nicht in älteren Browsern).

Ein anderes Beispiel wäre die Unterstützung von Mausclicks auf Bildern. Wenn Sie auf Mausclicks in Bildern mit einem JavaScript reagieren wollen, haben Sie das Problem, dass der Netscape Navigator das Click-Ereignis für `<img>`-Tags nicht unterstützt. Wenn Sie die Bilder aber in Anker-Elemente verpacken und für diese die Click-Ereignisse überwachen, erhalten Sie Code, der auch im Navigator funktioniert (siehe Listing 10.3).

Irgendwann werden Sie aber an die Grenzen dieser Tricks stoßen. Dann ist es an der Zeit, getrennten Code für die verschiedenen Browser aufzusetzen. Dazu müssen Sie aber erst einmal Code aufsetzen, der erkennt, in welchem Browser die Webseite gerade ausgeführt wird. Hierfür gibt es unterschiedliche Ansätze:

- Sie können den Browser anhand seiner Unterstützung bestimmter Objekte und Methoden erkennen:

```
<script type="text/javascript">
```

```

if (document.all) {
 // alert("Internet Explorer erkannt");
}
else if (document.layers) {
 // alert("Netscape Navigator erkannt");
}
else if (document.getElementById) {
 // alert("Netscape 6 erkannt");
}
else {
 // alert("Unbekannter Browser");
}
</script>

```

Dies macht vor allem dann Sinn, wenn Sie die betreffenden spezifischen Techniken des Browsers nutzen wollen. (Beachten Sie, dass es keine Sicherheit gibt, dass diese Unterscheidungen auch noch für zukünftige Browserversionen gelten. Würde der kommende Internet Explorer beispielsweise document.all nicht mehr unterstützen, würde er gemäß obigem Code als Netscape 6 eingestuft.)

- Sie können den Browser anhand seines Namens und seiner Version erkennen:

```

<script type="text/javascript">
// alles in Kleinbuchstaben umwandeln
var brw = navigator.appName.toLowerCase();
if (brw.indexOf("microsoft") != -1) {
 // alert("Internet Explorer erkannt");
}
else if (brw.indexOf("netscape") != -1) {
 if (parseInt(navigator.appVersion) < 5) {
 // alert("Netscape Navigator erkannt");
 }
 else {
 // alert("Netscape 6 erkannt");
 }
}
else {
 // alert("Unbekannter Browser");
}
</script>

```

- Sie können den Browser anhand seiner Browser-Maschine (MSIE, Mozilla, Gecko) erkennen. Ein sehr ausführliches Beispiel hierfür bietet der Quellcode der Webseite [http://developer.netscape.com/docs/examples/javascript/browser\\_type.html](http://developer.netscape.com/docs/examples/javascript/browser_type.html).

Die nächste Frage ist, was macht man in den Anweisungsblöcken der if-Verzweigung?

- Eine Möglichkeit ist, in den Anweisungsblöcken der if-Verzweigungen nur den Code einzuschließen, der browserspezifisch ist, und den allgemein gültigen Code vor beziehungsweise hinter der Verzweigung zu belassen.
- Wenn Sie jedoch an mehreren Stellen im Code eines <script>-Tags oder einer Funktion nach den Browsern verzweigen müssen, ist es einfacher und übersichtlicher, nur eine Verzweigung<sup>4</sup> zu haben und für jeden Browser den gesamten Code aufzusetzen.

- Wenn Sie in mehreren <script>-Tags oder Funktionen nach dem Browser verzweigen müssen, lohnt es sich, die eigentliche Browsererkennung nur einmalig im Header der Webseite durchzuführen und dabei einen Satz von Booleschen Variablen auf true oder false zu setzen. Später braucht man dann in der eigentlichen Verzweigung nur noch den Wert der Booleschen Variablen abzufragen und kann so Tipparbeit und Laufzeit sparen. (Das nachfolgende Beispiel demonstriert dies - obwohl es dort eigentlich unnötig ist, da in diesem Beispiel nur eine Verzweigung gebraucht wird.)
- Schließlich können Sie sich sogar überlegen, die Browsererkennung im Zuge des onload-Ereignisses der Eingangsseite zu Ihrem Web aufzusetzen und den Besucher danach via dynamischem HTML zu unterschiedlichen Webseiten zu führen.

### Listing 10.13: flaggen5.html - browserunabhängige Implementierung

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Flaggen</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
 // alles in Kleinbuchstaben umwandeln
 var brw = navigator.appName.toLowerCase();
 var ie = (brw.indexOf("microsoft") != -1);
 var nav = (brw.indexOf("netscape") != -1) &&
 (parseInt(navigator.appVersion) < 5);
 var net = (brw.indexOf("netscape") != -1) &&
 (parseInt(navigator.appVersion) >= 5);
 function loesen(str)
 {
 if (str == "bld3")
 document.getElementById(str).src = "richtig.gif";
 else
 document.getElementById(str).src = "falsch.gif";
 }
</script>
</head>
<body style="background-color: #F0F0F0" onload="init()">
<h1>Welche der folgenden Flaggen ist die Flagge von Sri Lanka?</h1>
<script type="text/javascript">
// Je nach Browser unterschiedlichen HTML-Code ausgeben
if (ie || net)
{
 document.write("\
<table>\
 <tr>\
 <td></td>\
 <td></td>\
 <td></td>\
 <td></td>\
 </tr>\
</table>\
");
}
else if (nav)
{
```

```

document.write("\
<table>\
<tr>\
 <td>\
 </></td>\
 <td>\
 </></td>\
 <td>\
 </></td>\
 <td>\
 </></td>\
</tr>\
</table>\
");
}
</script>

<p>Klicken Sie einfach auf die Flagge.</p>
</body>
</html>

```



*Beachten Sie, dass die doppelten Anführungszeichen in den document.write-Ausgaben mit Hilfe des vorangehenden \-Zeichens als einfache auszugebende Zeichen gekennzeichnet wurden. Ansonsten würde der Interpreter nämlich beim ersten Anführungszeichen annehmen, dass dort der auszugebende String endet.*

## 10.6 Rollover-Effekte für Schalter

Sicherlich sind Ihnen beim Surfen im Web auch schon Webseiten mit diesen herrlichen grafischen Schaltflächen begegnet, die aufleuchten, wenn man die Maus über sie bewegt, und vielleicht haben Sie sich schon überlegt, wie sich dies realisieren lässt.

Nun, wenn Sie bisher nicht hinter das Geheimnis dieser Schaltflächen gekommen sind, dann...

... werden Sie es an dieser Stelle auch nicht tun, denn wir haben uns entschlossen, Ihnen die Implementierung von Rollover-Schaltflächen als Übungsaufgabe zu stellen (mit Lösung im Anhang dieses Buches).

Da die Implementierung von Rollover-Schaltflächen aber nicht ganz trivial ist, erhalten Sie hier einige Hinweise und Tipps.

- Für die Implementierung von Rollover-Schaltflächen müssen Sie zwei Ereignisse implementieren: onmouseover (die Maus wird über der Schaltfläche bewegt) und onmouseout (die Maus wird von der Schaltfläche herunter bewegt).
- Wenn Sie auch den Netscape Navigator unterstützen wollen, sollten Sie nicht die Ereignisse der <img>-Tags abfangen, sondern die Ereignisse der umliegenden Anker- Elemente.
- Damit sich der Effekt ohne allzu große Verzögerung einstellt, empfiehlt es sich die Bilder für den Effekt nicht erst dann vom Webserver zu holen, wenn die Maus über eine Schaltfläche bewegt wird.

Laden Sie stattdessen alle benötigten Bilder vorab per JavaScript. Erzeugen Sie für jedes benötigte Bild ein Image-Objekt (var bild1 = new Image();) und weisen Sie dessen src-Eigenschaft den URL der Bilddatei zu (bild1.src = "meingif.gif";).

## 10.7 Dynamisch aufklappbare Listen

Ein weiteres Element, das man immer häufiger auf Webseiten findet, sind dynamische hierarchische Listen und Menüs, deren untergeordnete Ebenen durch Mausaktionen ein- und ausgeblendet werden können.

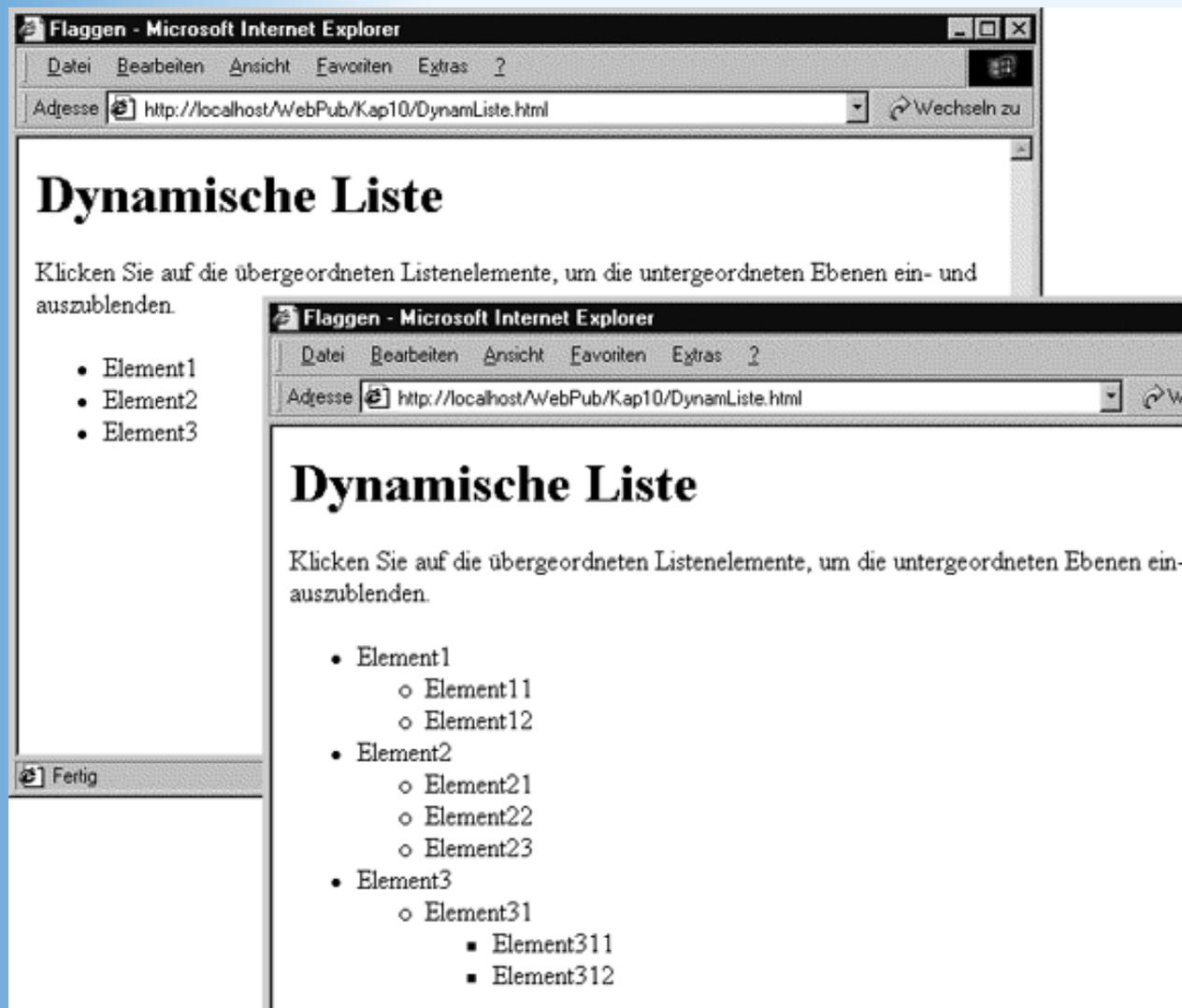


Abbildung 10.7: Zu- und aufgeklappte dynamische Liste

Das folgende Beispiel stellt Ihnen eine relativ einfache Variante zur Implementierung einer dynamischen Liste (genauer gesagt einer Aufzählung) vor. Die Implementierung unterstützt sowohl den Internet Explorer 5 als auch den Netscape 6-Browser. In anderen Browsern, wie zum Beispiel dem Netscape Navigator, wird die Liste stets vollständig aufgeklappt dargestellt.

### Listing 10.14: DynamListe.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Flaggen</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
```

```

</head>
<script type="text/javascript">
function initialisiereListe()
{
var src = document.getElementById("dynListe");
var loop1, loop2;
var kind1, kind2;
for(loop1 = 0; loop1 < src.childNodes.length; ++loop1)
{
kind1 = src.childNodes[loop1];

if(kind1.nodeName == "LI")
{
for(loop2 = 0; loop2 < kind1.childNodes.length; ++loop2)
{
kind2 = kind1.childNodes[loop2];
if (kind2.nodeName == "UL")
{
kind2.style.display = "none";
}
}
}
}
}

function dynListe(e)
{
var src = null;
if(document.all) // Internet Explorer, Version vernachlässigen wir
{
src = event.srcElement;
}
else if(document.getElementById) // Netscape 6
{
src = e.target;
}

var loop;
// Netscape 6 liefert Textelement bei Klick in Liste,
// daher in DOM-Hierarchie eine Ebene hochwandern
if (src.nodeName == "#text")
src = src.parentNode;
for(loop = 0; loop < src.childNodes.length; ++loop)
{
var kind = src.childNodes[loop];
if (kind.nodeName == "UL")
{
kind.style.display =
(kind.style.display == "none" ? "" : "none");
}
}
}
}

</script>
</head>
<body onload="initialisiereListe()">

```

```

<h1>Dynamische Liste</h1>
<p>Klicken Sie auf die übergeordneten Listenelemente, um die
untergeordneten Ebenen ein- und auszublenden.</p>
<ul id="dynListe" onclick="dynListe(event)">
 Element1

 Element11
 Element12

 Element2

 Element21
 Element22
 Element23

 Element3

 Element31

 Element311
 Element312

</body>
</html>

```

Schauen Sie sich zuerst den HTML-Code der Liste an. Die erste Ebene enthält drei Elemente, die jeweils wieder untergeordnete Listen enthalten (einmal mit zwei, einmal mit drei Elementen und einmal mit nur einem Element). Unter dem letzten Element gibt es noch eine dritte Ebene, das heißt, das Element der untergeordneten Ebene (Element31) enthält selbst auch wieder eine untergeordnete Ebene (mit zwei Elementen 311 und 312).

Alle Klickereignisse für die Liste werden in der obersten Listenebene abgefangen:

```

<ul id="dynListe" onclick="dynListe(event)">

```

Um in der Ereignisfunktion `dynListe()` feststellen zu können, welches Listenelement genau angeklickt wurde, müssen wir daher das globale `event`-Objekt bemühen. Da wir neben dem Internet Explorer 5 auch den Netscape 6-Browser unterstützen wollen, bedeutet dies, dass wir das Schlüsselwort `event` an die Ereignisfunktion übergeben.

Schauen wir uns nun den Code der Ereignisfunktion `dynListe()` an.

Nehmen wir an, der Besucher der Webseite hat irgendwo in die Liste geklickt. Unsere erste Aufgabe besteht dann darin, festzustellen, ob er auf ein Listenelement geklickt hat, zu dem es eine untergeordnete Ebene gibt (die dann ein- oder ausgeblendet werden soll). Die Information, welches Listenelement angeklickt wurde, erhalten wir von dem globalen `event`-Objekt. Da dieses für Internet Explorer und Netscape 6-Browser unterschiedlich implementiert ist (siehe Abschnitt 10.2), müssen wir eine `if`-

Verzweigung aufsetzen (siehe auch Abschnitt 10.5):

```
function dynListe(e)
{
 var src = null;
 if(document.all) // Internet Explorer, Version vernachlässigen wir
 {
 src = event.srcElement;
 }
 else if(document.getElementById) // Netscape 6
 {
 src = e.target;
 }
 ...
}
```

Im Internet Explorer enthält src jetzt das Listenelement, auf das der Besucher geklickt hat. Im Netscape 6-Browser enthält src dagegen das Textelement des Listenelements. Im Netscape 6-Browser müssen wir daher in der DOM-Hierarchie eine Ebene hochgehen (vom Textelement zum Listenelement).

```
...
var loop;
// Netscape 6 liefert Textelement bei Klick in Liste,
// daher in DOM-Hierarchie eine Ebene hochwandern
if (src.nodeName == "#text")
 src = src.parentNode;
...
```

Jetzt gehen wir alle untergeordneten Elemente des angeklickten Listenelements durch (siehe auch Abschnitt 10.4.5). Treffen wir dabei auf eine untergeordnete Aufzählung (nodeName == "UL") blenden wir diese ein oder aus.

```
for(loop = 0; loop < src.childNodes.length; ++loop)
{
 var kind = src.childNodes[loop];
 if (kind.nodeName == "UL")
 {
 kind.style.display =
 (kind.style.display == "none" ? "" : "none");
 }
}
```

Die Anweisung zum Ein- und Ausblenden der untergeordneten Listen müssen wir noch näher besprechen. Wir nutzen hier eine Stileigenschaft, die bisher noch nicht angesprochen wurde: display. Die Stileigenschaft display kann die Werte block, inline, list-item und none annehmen und weist den Browser an, nach welchem Verfahren er das Element anzeigen soll. Üblicherweise sollte man den Wert dieser Stileigenschaft nicht ändern, aber hier können wir uns zunutze machen, dass man das Element durch Zuweisung von "none" verschwinden lassen kann - und zwar so, dass keine Lücke zurückbleibt.

Wird die Ebene augenblicklich angezeigt, wollen wir sie ausblenden, ist sie ausgeblendet, soll sie eingeblendet werden. Für diese »Umschaltung« könnte man eine normale if-else- Verzweigung aufsetzen, man kann aber auch den Bedingungsoperator ?: nutzen. Der Ausdruck

```
kind.style.display == "none" ? "" : "none"
```

bedeutet:

»Wenn kind.style.display gleich "none" ist (?), dann liefere als Ergebnis des Ausdrucks "" zurück, ansonsten (:) liefere "none" zurück«

Zum guten Schluss wollen wir noch dafür sorgen, dass die Liste am Anfang zusammengeklappt ist. Dazu weisen wir der obersten Liste eine id zu und verbinden das onload-Ereignis des <body>-Tags mit einer Funktion, die die untergeordneten Ebenen der Liste ausblendet:

```
<body onload="initialisiereListe()">
```

```
...
<ul id="dynListe" ...">
```

In der Funktion `initialisiereListe()` greifen wir mit Hilfe der Methode `getElementById()` über die ID `dynListe` auf die Liste zu. Dann durchsuchen wir die Kindknoten der übergeordneten Liste nach Knoten vom Typ `LI` (Listenelemente). In einer zweiten Schleife werden diese wiederum daraufhin untersucht, ob sie untergeordnete Aufzählungen (`UL`) enthalten. Wenn ja werden diese ausgeblendet.

```
function initialisiereListe()
{
 var src = document.getElementById("dynListe");
 var loop1, loop2;
 var kind1, kind2;
 for(loop1 = 0; loop1 < src.childNodes.length; ++loop1)
 {
 kind1 = src.childNodes[loop1];

 if(kind1.nodeName == "LI")
 {
 for(loop2 = 0; loop2 < kind1.childNodes.length; ++loop2)
 {
 kind2 = kind1.childNodes[loop2];
 if (kind2.nodeName == "UL")
 {
 kind2.style.display = "none";
 }
 }
 }
 }
}
```



*Wenn Sie die dynamische Liste noch ein wenig bedienerfreundlicher machen wollen, verwenden Sie Plus- und Minuszeichen als Aufzählungssymbole und setzen Sie diese so, dass der Besucher erkennen kann, ob es zu einem Listenelement eine untergeordnete Ebene gibt oder nicht.*

## 10.8 Zusammenfassung

Heute haben wir wieder einmal viel geleistet und ein großes, aber auch überaus wichtiges Themengebiet behandelt: die Ereignisbehandlung und den Zugriff auf HTML-Elemente mittels JavaScript. Dass dieses Thema so kompliziert ist, liegt aber nicht daran, dass die dahinter stehenden Konzepte so schwer zu begreifen wären, sondern hat vielmehr damit zu tun, dass es in diesem Bereich so viele Unterschiede in den gängigen Browsern gibt. Was sollten Sie also von diesem Tag mitnehmen?

Erstens: Die grundlegende Ereignisbehandlung (JavaScript-Funktion als Wert an HTML- Ereignisattribut zuweisen) ist dabei in allen wichtigen Browsern gleich. Zu beachten ist lediglich, dass der Netscape Navigator 4 nicht alle im Standard vorgesehenen Ereignis/Tag- Kombinationen unterstützt. Manchmal kann man diese Defizite aber umgehen - wie im Falle der Klickereignisse für Bilder.

Zweitens: Die Modelle für die globale Ereignisbehandlung und die Verwendung des event-Objekts sind recht unterschiedlich. Andererseits sind sich zumindest der Internet Explorer und der Netscape 6-Browser hinsichtlich der globalen Ereignisbehandlung recht ähnlich und die Informationen im event-Objekt kann man manchmal durch die Übergabe passender Argumente an die Ereignisbehandlungsfunktionen (beispielsweise this-Objekt) ersetzen.

Drittens: Wenn Sie auf HTML-Elemente zugreifen, verwenden Sie möglichst nicht die proprietären Objekte `document.all` oder `document.layers` und auch nicht die an diese Objekte geknüpften Techniken (Zugriff über `id`). Nutzen Sie die allgemein unterstützten Array-Eigenschaften des `document`-Objekts (`document.images[]`, `document.forms[]`, etc.) sowie die Methoden und Techniken, die im DOM-Standard definiert sind. Letztere werden derzeit zwar nur von den aktuellsten Browsern unterstützt, doch liegt in Ihnen zweifelsohne die Zukunft.

## 10.9 Fragen und Antworten

**Frage:**

**Ich habe für einen `<p>`-Absatz und eine eingeschlossene `<b>`-Textpassage jeweils das `onclick`-Ereignis abgefangen. Wenn ich beiden Ereignisfunktionen das Schlüsselwort `this` übergebe, welche Objekte repräsentiert es in den beiden Funktionen? Wenn ich in beiden Funktionen über das event-Objekt den Ursprung des Ereignisses abfrage (`event.srcElement` bzw. `event.target`), welche Objekte erhalte ich in den beiden Funktionen?**

*Antwort:*

*Das `this`-Objekt repräsentiert immer das Objekt, dessen `onclick`-Ereignisfunktion es übergeben wurde. Das Ursprungsobjekt, das über `event` abgefragt werden kann, repräsentiert dagegen immer das innerste Objekt, in dem das Ereignis ursprünglich ausgelöst wurde.*

**Frage:**

**In Netscape-Browsern kann man durch Aufruf von `window.captureEvents()` Ereignisse global abfangen lassen. Kann man das globale Abfangen eines Ereignisses auch wieder abschalten?**

*Antwort:*

*Ja! Verwenden Sie hierzu die Methode `window.releaseEvents()`.*

**Frage:**

**Kann man in einem `captureEvents`-Aufruf mehrere Ereignisse angeben?**

*Antwort:*

*Ja! Sie müssen die Ereignistypen lediglich ODER-verknüpfen.*

```
window.captureEvents(Event.CLICK | Event.MOUSEDOWN);
```

### Frage:

**Wie kann ich beim Navigieren mit den DOM-Knoteneigenschaften `firstChild`, `lastChild`, etc. sicher sein, dass der Knoten, zu dem ich wechsele, auch existiert?**

### Antwort:

*Wenn der Knoten nicht existiert, liefern die betreffende Knoteneigenschaften - und auch die Methode `getElementById()` - den Wert null zurück. Da JavaScript den Nullwert als Booleschen Wert `false` interpretiert, kann man leicht mit einer `if`-Anweisung überprüfen, ob ein Knoten existiert oder nicht:*

```
var p1 = document.getElementById("absatz1");
if(!p1)
 {
 alert("kein p1");
 }
else
 {
 p1.style.color = "red";
 var p1Kinder = p1.childNodes;
 if (!p1Kinder[1])
 {
 alert("kein p1Kinder[1]");
 }
 else
 {
 p1Kinder[1].style.backgroundColor = "red";
 }
 }
```

Beachten Sie, dass die Eigenschaft `childNodes` im Falle eines Scheiterns keinen Nullwert, sondern ein leeres Knoten-Array zurückliefert. Wenn Sie aber über die Indizierung auf ein nicht vorhandenes Array-Element zugreifen wollen, erhalten Sie `null` zurück.

## 10.10 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

1. Zählen Sie aus dem Gedächtnis ein paar JavaScript-Ereignisse auf.
2. Wie greift man auf das globale event-Objekt zu? Wie greift man auf das auslösende HTML-Element oder die Koordinaten eines Mausklicks zu?
3. Auf das wievielte Bild einer Webseite greifen Sie über `document.images[2]` zu?
4. Mit welcher DOM-Methode kann man durch Angabe des id-Wertes auf HTML-Elemente zugreifen? Welche Browser unterstützen diese Methode?
5. Wie kann man die Stileigenschaften eines Objekts ändern?
6. Wie kann mit JavaScript erkennen, in welchem Browser die Webseite gerade angezeigt wird?

# Übungen

1. Implementieren Sie eine Tabelle mit drei grafischen Rollover-Schaltflächen.

---

❖ Kapitel Inhalt Index **SAMS** ❖ Top Kapitel❖

---

1  
DOM ist eine Abkürzung für Document Object Model.

2  
Ob die Marktentwicklung der letzten Jahre jedoch allein auf diese objektiven Vorteile zurückzuführen ist, sei dahingestellt.

3  
DOM ist eine Abkürzung für Document Object Model

4  
pro <script>-Tag beziehungsweise pro Funktion.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

## Tag 11

# JavaScript und Formulare

Heute kommen wir zu einem der ältesten und interessantesten Einsatzgebiete von JavaScript: der Bearbeitung von Steuerelement- und Formulareingaben.

Die Themen heute:

- Ereignisverarbeitung für Steuerelemente und Formulare
- Zugriff auf Steuerelemente
- Einsatz von Steuerelementen und Formularen zum Aufbau von Benutzeroberflächen
- Realisierung einer Eingabemaske
- Realisierung eines einarmigen Banditen
- Vollständigkeitsprüfung von Formularen
- Verifizierung von Benutzereingaben

## 11.1 Zugriff und Ereignisverarbeitung für Steuerelemente und Formulare

Wie man mit HTML Steuerelemente und Formulare definiert, haben Sie bereits in Kapitel 3 gesehen. Auch ein erstes Beispiel zur Ereignisbehandlung für Steuerelemente haben wir Ihnen dort gezeigt.

### Listing 11.1: schalter.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Schalter</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript">
</head>
<body>
<h2>Schalter als Beispiel für ein Steuerelement</h2>
<form>
<input type="button" name="schalter1"
 value="Klick mich!" onclick="alert('Treffer')">
</form>
</body>
</html>
```

Man sieht: die Ereignisbehandlung für Steuerelemente unterscheidet sich in keiner Weise von der Ereignisbehandlung für die anderen HTML-Tags. Alles, was wir jetzt noch wissen müssen, um Steuerelemente und Formulare sinnvoll und nutzbringend in Verbindung mit JavaScript einsetzen zu können, ist

- wie man von JavaScript-Code auf Steuerelemente in Webseiten zugreifen kann (beispielsweise um als Antwort auf ein Steuerelement-Ereignis den »Wert« des Steuerelements abzufragen oder zu verändern),
- welche Ereignisse für welche Steuerelemente unterstützt werden.

## Zugriff auf Steuerelemente

Selbstverständlich kann man auf Steuerelemente wie auf jedes andere HTML-Element zugreifen - also beispielsweise über die proprietären Objekte `document.all` (Internet Explorer) und `document.layers` (Netscape Navigator) oder über die im DOM-Standard vorgeschlagenen Methoden (`getElementById()`, etc.).

Einfacher ist jedoch der Zugriff über das `forms`-Array des `document`-Objekts. Voraussetzung ist natürlich, dass die Steuerelemente in `<form>`-Tags eingeschlossen sind - was man aber wegen des Netscape Navigators (der Steuerelemente nur in Formularen erlaubt) sowieso tun sollte. Zur Belohnung wird der Zugriff über das `forms`-Array von allen wichtigen Browsern unterstützt.

Die Verwendung der Array-Objekte von `document` dürfte Ihnen noch aus Kapitel 10 vertraut sein. Beim Laden der Webseite geht der Browser die einzelnen Formulare, die auf der Webseite definiert sind, durch und trägt sie in der Reihe ihrer Definition im HTML- Code in das Array `forms` ein. Danach können wir über einen nullbasierten Index auf die Formulare der Webseite zugreifen:

```
document.forms[0] // Zugriff auf das erste Formular einer Webseite
document.forms[1] // Zugriff auf das zweite Formular einer Webseite
```

und so weiter.



*Auf Formulare die einen Namen (`name`- oder `id`-Attribut) haben, kann man noch über andere Syntaxformen zugreifen - beispielsweise `document.forms["FormularName"]` oder `document.FormularName` oder einfach nur `FormularName`. Diese Syntaxformen werden allerdings nicht von allen Browsern gleichermaßen unterstützt.*

Alle Formulare verfügen über einen Satz von Eigenschaften, die wie üblich den HTML- Attributen des `<form>`-Tags entsprechen und die man abfragen oder verändern kann: `action`, `encoding`, `method`, `name`, `target`.

```
document.forms[0].target = "frameUnten";
```

Meist benötigen wir das Formular-Objekt jedoch nur, um auf die Steuerelemente im Formular zuzugreifen. Diese können über das elements-Array des Formular-Objekts angesprochen werden.

```
document.forms[0].elements[0].value = "Neuer Text"; // Zugriff auf
 // 1. Element in
 // 1. Formular
document.forms[0].elements[1].value = "Neuer Text"; // Zugriff auf
 // 2. Element in
 // 1. Formular
document.forms[1].elements[0].value = "Neuer Text"; // Zugriff auf
 // 1. Element in
 // 2. Formular
```

Das folgende Beispiel demonstriert, wie man per JavaScript-Code die Inhalte von Textelementen verändern kann.

### Listing 11.2: zugriff.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Steuerelemente</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript">
</head>
<body>
<h2>JavaScript-Zugriff auf Steuerelemente</h2>
<form>
<table>
 <tr>
 <td>
 <input type="text" name="text1" value="" />
 </td>
 </tr>
 <tr>
 <td>
 <input type="text" name="text2" value="" />
 </td>
 </tr>
</table>
</form>
<script type="text/javascript">
 document.forms[0].elements[0].value = "Hallo";
 document.forms[0].elements[1].value = "von Ihrem JavaScript";
</script>
</body>
</html>
```



**Abbildung 11.1: Manipulation von Steuerelementen über Javascript**

Wie man über JavaScript-Code auf Steuerelemente zugreift, haben wir nun gesehen. Jetzt müssen Sie nur noch wissen, welche Ereignisse und Eigenschaften für welche Steuerelemente definiert sind.

## Ereignisbehandlung für die verschiedenen Steuerelemente

Beginnen wir gleich mit dem ersten Typ von Steuerelement.

### Ein- und mehrzeilige Textfelder

```
<input type="text" name="T1" value="hier eintippen" size="20"
 maxlength="40" />
<input type="password" name="P1" />
<textarea name="S1" rows="2" cols="20">Inhalt</textarea/>
```

Ein- und mehrzeilige Textfelder unterstützen folgende Ereignisse:

- onclick, der Besucher hat mit der Maus in das Textfeld geklickt (wird nicht vom Navigator unterstützt).
- onfocus, das Steuerelement hat den Fokus erhalten (Durch Anklicken oder Drücken der (Tab)-Taste).
- onblur, das Steuerelement hat den Fokus verloren (der Websurfer hat zu einem anderen Steuerelement gewechselt).
- onchange, das Steuerelement hat den Fokus erhalten, nachdem zuvor sein Inhalt geändert wurde.
- onselect, im Steuerelement wurde ein Textbereich markiert (wird nicht vom Navigator unterstützt).

Über die Eigenschaft `value` können Sie den Inhalt des Steuerelements auslesen oder ändern. Die folgende Ereignisbehandlung könnte von einem Textelement zur Abfrage des Besuchernamens stammen:

```
<input type="text" name="name" value="Geben Sie Ihren Namen ein"
 size="20" maxlength="40" onchange="hallo(this)" />
...
function hallo(elem)
{
 var str = elem.value;
 elem.value = "Hallo " + elem.value;
}
```

Wenn Sie den Text eines mehrzeiligen Textfeldes (`<textarea>`) ändern, können Sie Zeilenumbrüche in Form des Sonderzeichens `\n` eingeben:

```
elem.value = "Erste Zeile \n Zweite Zeile";
```



*Über Javascript können Sie den Inhalt eines Textfeldes auch dann ändern, wenn das Attribut `readonly="readonly"` gesetzt ist! Ob aber das Ändern des Textes via JavaScript ein `onchange`-Ereignis auslöst, hängt vom Browser ab.*

Nicht ändern, aber abfragen können Sie den anfänglichen Inhalt des Steuerelements, der im HTML-Attribut `value` festgelegt wurde. Dieser ist in der Eigenschaft `defaultValue` gespeichert.

## Schalter

```
<input type="button" name="B1" value="Schaltfläche" />
```

Für Schalter gibt es nur drei Ereignisse: `onfocus`, `onblur` und `onclick`, von denen `onclick` natürlich das Interessanteste ist. Wir werden weiter unten noch einige Beispiele für die Verarbeitung von `onclick`-Ereignissen für Schalter sehen.

Über die Eigenschaft `value` kann man den Titel des Schalters abfragen oder neu setzen.

```
<input type="button" value="Klick mich!" name="B1"
 onclick="angeklickt(this)">
...
function angeklickt(elem)
{
 elem.value = "Danke!";
}
```

Für die beiden Schalter »Submit« und »Reset« zum Abschicken beziehungsweise Zurücksetzen des Formulars braucht man üblicherweise kein onclick-Ereignis zu definieren.

```
<input type="submit" name="B2" value="Abschicken" />
<input type="reset" name="B3" value="Zurücksetzen" />
```

Drückt der Besucher den Submit-Schalter eines Formulars, wird zuerst das onsubmit- Ereignis des Formulars ausgelöst. Wenn Sie in das Abschicken der Formulareingaben eingreifen wollen, ist dies die richtige Stelle. Setzen Sie eine Ereignisbehandlung für das onsubmit-Ereignis des Formulars auf und lassen Sie diese den Wert true zurückliefern, wenn die Formulareingaben zum Schluss an den im action-Attribut spezifizierten URL verschickt werden sollen, oder liefern Sie false zurück, wenn die Formulareingaben nicht verschickt werden sollen. Wenn Sie keine Ereignisbehandlung für onsubmit vorsehen, werden die Formulareingaben automatisch beim Klick des Submit-Schalters versendet.

Drückt der Besucher den Reset-Schalter eines Formulars, wird zuerst das onreset-Ereignis des Formulars ausgelöst und dann der Inhalt der Steuerelemente im Formular auf ihre Default-Werte zurückgesetzt. Wie im Falle des Submit-Schalters können Sie in die Ereignisbehandlung eingreifen, indem Sie eine Ereignisfunktion für das onreset-Ereignis des Formulars aufsetzen.



*Mehr zum Abschicken und Zurücksetzen von Formularen gleich in Abschnitt 11.3.*

## Kontrollkästchen und Optionsfelder

```
<input type="checkbox" name="C1" checked="checked" />
<input type="radio" name="R1" value="1" checked="checked" />
```

Kontrollkästchen und Optionsfelder sind letztlich nichts anderes als besondere Formen von Schaltern, für die die Ereignisse onfocus, onblur und onclick unterstützt werden.

Die interessanteste Eigenschaft für diese Steuerelemente ist nicht value, sondern checked. Über diese Eigenschaft können Sie feststellen, ob der Besucher das Element gesetzt hat, oder das Steuerelement via JavaScript-Code setzen oder löschen.

```
<input type="checkbox" name="rot" onclick="farbe(this)" />
...
function farbe(elem)
{
 if(elem.checked == true)
 {
 document.bgColor = "red";
 }
 else
```

```

 {
 document.bgColor = "white";
 }
 }

```

Etwas anders sieht die Programmierung mit gruppierten Optionsfeldern aus.

```

<form>
 <table border="0" cellspacing="5">
 <tr>
 <td><input type="radio" name="hintergrund" value="weiss"
 onclick="document.bgColor='white';" />Weiss</td>
 <td><input type="radio" name="hintergrund" value="rot"
 onclick="document.bgColor='red';" />Rot</td>
 <td><input type="radio" name="hintergrund" value="blau"
 onclick="document.bgColor='blue';" />Blau</td>
 </tr>
 </table>
</form>

```

Wenn Sie mehreren Optionsfeldern den gleichen Namen zuteilen, bilden diese eine Gruppe.



*Gruppierte Optionsfelder werden immer dann eingesetzt, wenn der Besucher zwischen verschiedenen Optionen wählen soll, die sich gegenseitig ausschließen.*

Das Besondere an den gruppierten Optionsfeldern ist, dass von den Optionsfeldern einer Gruppe immer nur eines ausgewählt sein kann. Außerdem kann man Optionsfelder nicht deselektieren, das heißt, wenn der Besucher auf ein Optionsfeld klickt, wird dieses auf jeden Fall gesetzt und alle andere Optionsfelder der Gruppe werden gelöscht. Dies vereinfacht die Behandlung von onclick-Ereignissen, da man nicht erst feststellen muss, ob das Optionsfeld durch den Mausklick gesetzt oder gelöscht wurde:

```

<input type="radio" name="hintergrund" value="rot"
 onclick="document.bgColor='red';" />

```

Etwas komplizierter wird es, wenn man - statt für jedes Optionsfeld eine eigene Ereignisbehandlung aufzusetzen (siehe oben) - eine JavaScript-Funktion zur Behandlung aller Optionsfelder einer Gruppe oder gar zur Behandlung der gesamten Formulareingaben schreibt. Dann stellt sich nämlich meist das Problem, wie man die einzelnen Optionsfelder einer Gruppe identifiziert. Über die Namen der Optionsfelder ist dies nicht möglich, da ja alle Optionsfelder einer Gruppe den gleichen Namen tragen. Also weicht man auf den value-Wert aus und teilt allen Optionsfeldern in einer Gruppe eigene value-Werte zu, über die man die einzelnen Optionsfelder im JavaScript-Code identifizieren kann. Dies könnte dann beispielsweise wie folgt aussehen:

### Listing 11.3: Auszug aus optionen.html

```
<form>
 <table border="0" cellspacing="5">
 <tr>
 <td><input type="radio" name="hintergrund" value="weiss"
 onclick="farbe(this)" />Weiss</td>
 <td><input type="radio" name="hintergrund" value="rot"
 onclick="farbe(this)" />Rot</td>
 <td><input type="radio" name="hintergrund" value="blau"
 onclick="farbe(this)" />Blau</td>
 </tr>
 </table>
</form>
```

Hier werden drei Optionsfelder angeboten, über die man eine Farbe für den Hintergrund der Webseite auswählen kann. Alle drei Optionsfelder haben den gleichen Namen und verwenden die gleiche Ereignisfunktion für das onclick-Ereignis. Eindeutig identifiziert werden sie über ihre value-Werte.

In der Ereignisbehandlungsfunktion kann man den value-Werte des aktuell angeklickten Optionsfeldes abfragen und danach entscheiden, wie der Hintergrund der Webseite einzufärben ist.

### Listing 11.4: Auszug aus optionen.html

```
<script type="text/javascript">
function farbe(elem)
{
 switch(elem.value)
 {
 case "weiss": document.bgColor = "white"; break;
 case "rot": document.bgColor = "red"; break;
 case "blau": document.bgColor = "blue"; break;
 }
}
</script>
```



*Man hätte die Hintergrundfarbe natürlich auch gleich beim Aufruf der Funktion übergeben können (siehe Übung 1) doch ging es uns ja weniger um eine effiziente Implementierung als vielmehr um die Identifizierung der einzelnen Optionsfelder aus einer Gruppe. Ein weiteres Beispiel hierzu finden Sie übrigens in Kapitel 13, Übung 1.*

# Ereignisbehandlung für Auswahllisten

```
<select name="obst" size="5" multiple="multiple">
 <option>Bananen</option>
 <option selected="selected">Äpfel</option>
 <option selected="selected">Orangen</option>
 <option>Kirschen</option>
 <option>Kiwis</option>
 <option>Stachelbeeren</option>
 <option>Pfirsiche</option>
</select>
```

Bei der Programmierung mit Auswahllisten geht es meist darum,

- festzustellen, welche Optionen ausgewählt sind,
- die anfänglich vorgegebene Auswahl wiederherzustellen,
- neue Optionen aufzunehmen oder,
- eingetragene Optionen zu löschen.

Dazu muss man wissen, wie man auf die einzelnen Optionen einer Auswahlliste zugreifen kann. Vermutlich ahnen Sie es schon. Richtig, für jedes Auswahllisten-Objekt gibt es wieder ein Array, das in diesem Falle `options` heißt, über das man auf die einzelnen Optionen zugreifen kann.

Wie geht man also vor, wenn man ermitteln möchte, welche Optionen der Besucher in einer Auswahlliste ausgewählt hat?

## Ausgewählte Optionen herausfiltern

Erlaubt die Auswahlliste nur die Auswahl eines einzigen Elements kann man den Index dieses Elements direkt der Eigenschaft `selectedIndex` des Auswahllistenobjekts entnehmen:

```
var ausgewaehlt = document.forms[n].elements[m].selectedIndex;1
```

Erlaubt die Auswahlliste allerdings wie in obigem Beispiel die Auswahl mehrerer Elemente, liefert `selectedIndex` nur den Index des ersten ausgewählten Elements. Meist programmiert man für Auswahllisten mit Mehrfachauswahl daher eine Schleife, in der man die einzelnen Optionen durchgeht und nachschaut, welche der Optionen ausgewählt sind (`options[]`-Eigenschaft `selected`).

### Listing 11.5: auswahl.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Auswahllisten</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
```

```

function auswahl()
{
var auswahl = document.forms[0].elements[0];
for(i = 0; i < auswahl.options.length; i++)
 if(auswahl.options[i].selected == true)
 alert(auswahl.options[i].text);
}
</script>
</head>
<body>
<h2>Auswahllisten</h2>
<table border="0" width="100%" cellspacing="5">
 <form>
 <tr>
 <td>
 <select name="obst" size="5" multiple="multiple">
 <option>Bananen</option>
 <option selected="selected">Äpfel</option>
 <option selected="selected">Orangen</option>
 <option>Kirschen</option>
 <option>Kiwis</option>
 <option>Stachelbeeren</option>
 <option>Pfirsiche</option>
 </select>
 </td>
 </tr>
 <tr>
 <td>
 <input type="button" value="Anzeigen" onclick="auswahl()" />
 </td>
 </tr>
 </form>
</table>
</body>
</html>

```

Dieses Beispiel ist so konstruiert, dass der Besucher erst in Ruhe seine Auswahl treffen kann und danach den Anzeigen-Schalter anklickt. Als Antwort auf das onclick-Ereignis des Schalters werden dann die Titel der ausgewählten Optionen (text-Eigenschaft) nacheinander mittels alert() ausgegeben.



*Statt die Auswertung der ausgewählten Elemente mit dem Klicken eines eigenen Schalters zu verbinden, hätte man auch das onblur-Ereignis der Auswahlliste*

abfangen können.

## Wie kann man die im HTML-Code vorgegebene Auswahl wiederherstellen?

Ob für eine Option das HTML-Attribut `selected` gesetzt ist oder nicht, kann man am Wert der `options[]`-Eigenschaft `defaultSelected` ablesen. Mit diesem Wissen ist es nicht mehr schwer, die Ereignisfunktion angeklickt aus Listing 11.5 so umzuschreiben, dass beim Klick auf den Schalter die Auswahl des Benutzers verworfen und die HTML-Auswahl wieder hergestellt wird.

```
function angeklickt()
{
 var auswahl = document.forms[0].elements[0];
 for(i = 0; i < auswahl.options.length; i++)
 if(auswahl.options[i].defaultSelected == true)
 auswahl.options[i].selected = true;
 else
 auswahl.options[i].selected = false;
}
```

## Wie kann man Optionen löschen?

Um eine Option zu löschen, braucht man nur das entsprechende Array-Element auf null zu setzen.

```
function loeschen()
{
 var auswahl = document.forms[0].elements[0];
 // markierte Optionen löschen
 for(i = 0; i < auswahl.options.length; i++)
 if(auswahl.options[i].selected == true)
 auswahl.options[i] = null;
}
```

## Wie kann man neue Optionen einfügen?

Um eine Option neu einzufügen, erzeugt man ein neues Objekt der Klasse `Option` und weist dieses Objekt einem Element im `options`-Array zu. Je nachdem wie man den Index für `options[]` wählt, kann man ein neues Element hinten anhängen oder ein bestehendes Element ersetzen.

```
function einfuegen()
{
 var auswahl = document.forms[0].elements[0];
 // Titel für neue Option abfragen
 var titel = prompt("Wie soll die neue Option heißen?");
 // Option ans Ende der Liste anhängen
 var neu = new Option(titel);
 auswahl.options[auswahl.length] = neu;
}
```

## 11.2 Benutzeroberflächen

Auch wenn wir unsere Steuerelemente aus Rücksicht auf den Navigator-Browser immer in `<form>`-Tags kleiden, muss man diese »Formulare« nicht unbedingt zur Übertragung von Benutzereingaben an den Webserver nutzen. Man kann sie auch zum Aufbau grafischer Benutzeroberflächen verwenden. Das Tolle dabei ist, dass die gesamte Funktionalität hinter den Oberflächenelementen mit JavaScript realisiert werden kann.

Als praktisches Beispiel für den Einsatz von Steuerelementen zum Aufbau grafischer Oberflächenelemente stellen wir Ihnen hier einen kleinen einarmigen Banditen vor.

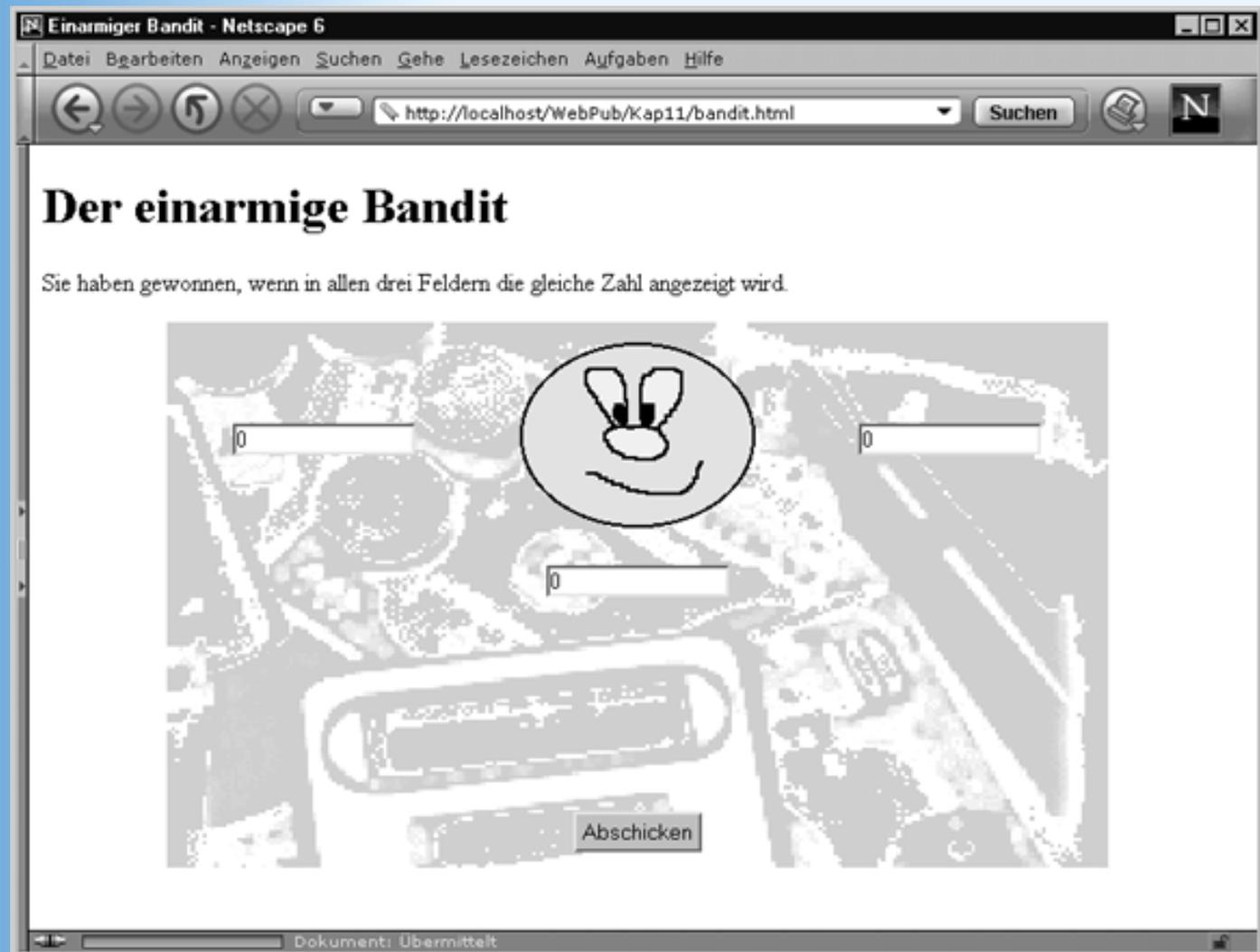


Abbildung 11.2: JavaScript-unterstützte grafische Benutzeroberfläche

### Listing 11.6: bandit.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Einarmiger Bandit</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
```

```

<script type="text/javascript">
var getroffen = true;
var bilder = new Array(2);
 bilder[0] = new Image();
 bilder[0].src = "glueck.gif";
 bilder[1] = new Image();
 bilder[1].src = "pech.gif";
function treffer()
{
 with (document.forms[0])
 {
 if (elements[0].value == elements[1].value &&
 elements[1].value == elements[2].value)
 {
 getroffen = true;
 document.images[0].src = bilder[0].src;
 }
 }
}
function zufall()
{
 if(getroffen == true)
 {
 getroffen = false;
 document.images[0].src = bilder[1].src;
 }
 for(var count = 0; count <= 2; count++)
 {
 var zufallszahl = Math.random() * 4;
 // Umwandlung der Gleitkommazahl in eine Ganzzahl
 // Wegen dem Navigator müssen wir Werte < 1 gesondert
 // behandeln
 if (zufallszahl > 1)
 zufallszahl = parseInt(zufallszahl);
 else
 zufallszahl = 0;
 var zufallszahl = parseInt(Math.random() * 4);
 document.forms[0].elements[count].value = zufallszahl;
 }
 treffer();
}
</script>
</head>
<body>
<h1>Der einarmige Bandit</h1>
<p>Sie haben gewonnen, wenn in allen drei Feldern
 die gleiche Zahl angezeigt wird.</p>
<form>
 <table width="600" height="350" border="0" cellspacing="5"

```

```

 cellpadding="5"
 style="background-image: url('hintergrund.gif');
 margin-left:5em">
<tr>
 <td width="200" align="center"><input type="text" name="T1"
 -size="15" readonly="readonly" /></td>
 <td width="200" align="center"></td>
 <td width="200" align="center"><input type="text" name="T3"
 -size="15" readonly="readonly" /></td>
</tr>
<tr>
 <td> </td>
 <td width="200" align="center"><input type="text" name="T2"
 -size="15" readonly="readonly" /></td>
 <td> </td>
</tr>
<tr>
 <td> </td>
 <td width="200" height="150" align="center" valign="bottom"><input
 -type="button" value="Abschicken" name="B1"
 -onclick="zufall()"></td>
 <td> </td>
</tr>
</table>
</form>
</body>
</html>

```

Jedes Mal, wenn der Spieler auf den **Abschicken**-Schalter klickt, wird die Funktion `zufall()` aufgerufen. Diese prüft, ob das letzte Spiel gewonnen worden ist. Wenn ja (getroffen gleich `true`), setzt sie die Boolesche Variable `getroffen` auf `false` zurück und lädt das Pech-Smiley. Dann zieht sie drei neue Zufallszahlen und gibt diese in die Textfelder der grafischen Oberfläche aus. Zum Schluss ruft sie die Funktion `treffer()`, die feststellen soll, ob das aktuelle Spiel gewonnen wurde.

Die Funktion `treffer()` prüft, ob alle drei Textfelder den gleichen Wert enthalten, wenn ja wird die Boolesche Variable `getroffen` auf `false` gesetzt und das Glücks-Smiley angezeigt.

## 11.3 Formulareingaben

Ein typisches Einsatzgebiet für JavaScript in Verbindung mit »echten« Formularen ist die Vollständigkeitsprüfung und gegebenenfalls die Verifizierung der Formulareingaben.

### Vollständigkeitsprüfung von Formularen

Formulare, die dazu dienen, Daten vom Browser an den Server der Webseite zurückzuschicken, weisen gegenüber den »Schein«-Formularen zur Gestaltung von grafischen Oberflächen drei Besonderheiten auf:

- Im <form>-Tag wird über das action-Attribut angegeben, an welche URL die Daten zu schicken sind.
- Ebenfalls im <form>-Tag kann auch das method-Attribut spezifiziert werden, das festlegt, auf welche Weise die Daten an das verarbeitende Programm übergeben werden sollen (GET oder POST, siehe Kapitel 17).
- Am Ende des Formulars wird ein Abschicken-Schalter (type="submit") eingerichtet (gegebenenfalls auch ein Zurücksetzen-Schalter (type="reset")).

Drückt der Besucher den Abschicken-Schalter eines Formulars, wird zuerst das onsubmit-Ereignis des Formulars ausgelöst. Ist keine Ereignisbehandlungsfunktion für dieses Ereignis eingerichtet, fährt der Browser damit fort, dass er die Daten an den im action-Attribut spezifizierten URL schickt.

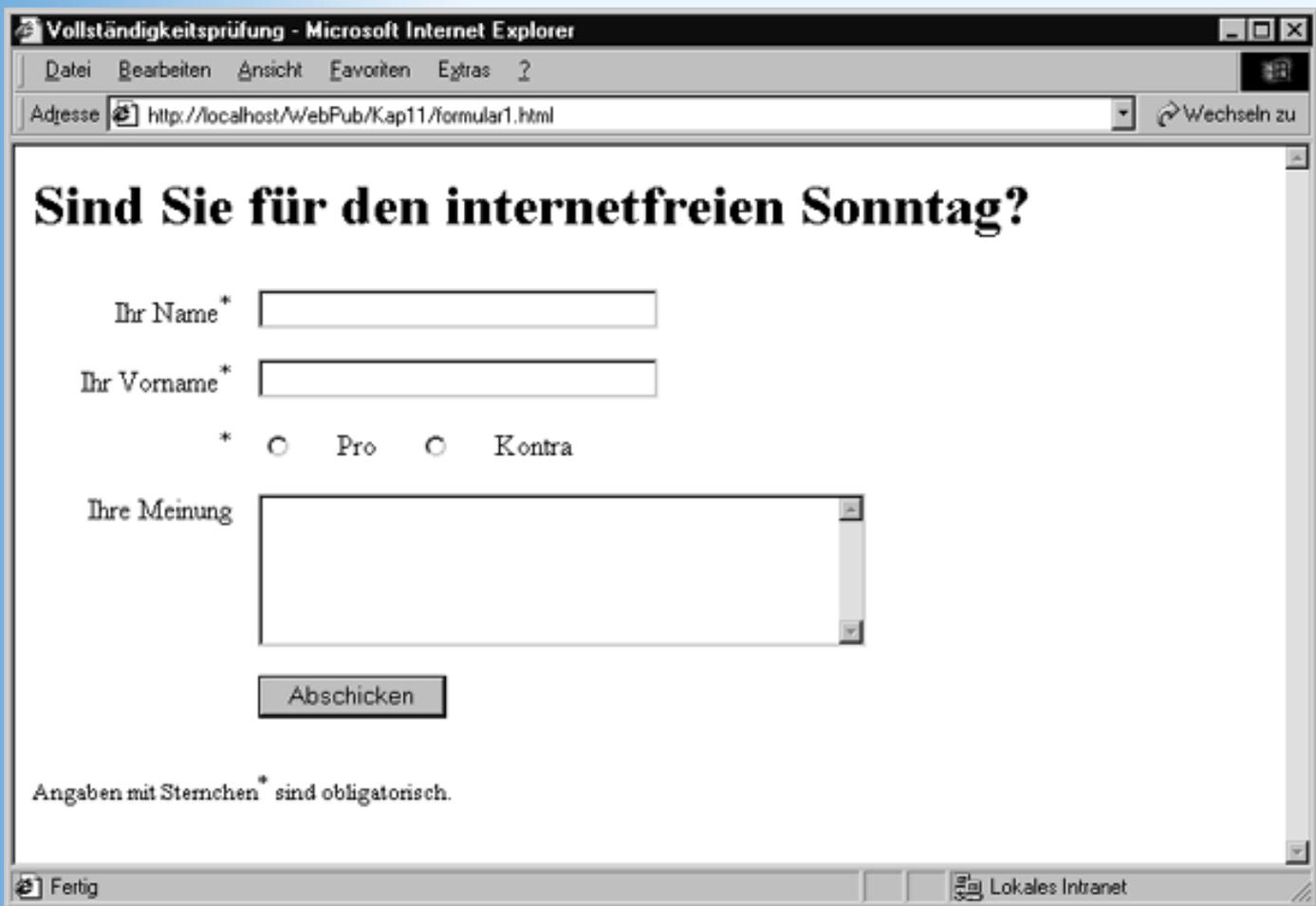
Das Problem dabei ist, dass viele Websurfer die Formulare aus Faulheit, Desinteresse oder zur Wahrung Ihrer Privatsphäre zuerst einmal nur sehr unvollständig ausfüllen. Nun kann man dies natürlich in dem Programm, das die Formulardaten auf Seiten des Webserverns entgegen nimmt, feststellen und eine entsprechende Aufforderung, das Formular doch bitte vollständig auszufüllen, zurückschicken. Wegen der Übertragung der Daten über das Netz ist dieses Verfahren jedoch sehr zeitaufwendig und unökonomisch. Effizienter ist es meist, die Vollständigkeit eines Formulars mit Hilfe von JavaScript bereits auf der Clientseite zu überprüfen.

Dazu müssen Sie eine Ereignisbehandlungsfunktion für das onsubmit-Ereignis des Formulars aufsetzen. In dieser prüfen Sie, ob alle relevanten Felder des Formulars ausgefüllt wurden. Wenn ja, liefert die Funktion true zurück. Wurde dagegen auch nur ein wichtiges Feld ausgelassen, wird die Funktion mit der Rückgabe von false beendet. Den Rückgabewert dieser Funktion erklärt man dann zum Rückgabewert von onsubmit:

```
onsubmit="return ereignisFunktion()"
```

Dies ist wichtig, da nur so der Browser erkennen kann, ob die Formulardaten nach Ausführung der Funktion gesendet (onsubmit = true) oder verworfen werden sollen (onsubmit = false).

Schauen Sie sich einmal Abbildung 11.3 an.



**Abbildung 11.3: Umfrage-Formular**

Der Code, der dieses Formular erzeugt, sieht wie folgt aus.

**Listing 11.7: Auszug aus formular1.html**

```
<body>
<h1>Sind Sie für den internetfreien Sonntag?</h1>
<form method="post" action="http://server.com/cgi-bin/programm.pl"
 onsubmit="return vollstaendig()">
 <table border="0" width="500" cellspacing="5" cellpadding="5">
 <tr>
 <td width="150" align="right">Ihr Name[*]</td>
 <td width="350" colspan="4"><input type="text" name="T1"
 size="30" /></td>
 </tr>
 <tr>
 <td width="150" align="right">Ihr Vorname[*]</td>
 <td width="350" colspan="4"><input type="text" name="T2"
 size="30" /></td>
 </tr>
 <tr>
 <td width="150" align="right">[*]</td>
 <td width="30"><input type="radio" value="pro" name="R1" /></td>
 <td width="30">Pro</td>
```

```

 <td width="30"><input type="radio" value="kontra"
 name="R1" /></td>
 <td width="260">Kontra</td>
</tr>
<tr>
 <td width="150" align="right" valign="top">Ihre Meinung</td>
 <td width="350" colspan="4"><textarea name="T3" rows="5"
 cols="40"></textarea></td>
</tr>
<tr>
 <td width="150"> </td>
 <td width="350" colspan="4"><input type="submit" name="senden"
 value="Abschicken" /></td>
</tr>
</table>
</form>
<p style="font-size: 0.8em">Angaben mit Sternchen[*] sind
obligatorisch.</p>
</body>

```

Interessant ist an diesem Code für uns im Moment nur der Abschicken-Schalter (`type="submit"`)<sup>2</sup> am Ende der Tabelle und die Deklaration des `<form>`-Tags mit den Angaben des Ziel-URLs und der `onsubmit`-Ereignisbehandlung.

Tritt das `onsubmit`-Ereignis ein, wird die Funktion `vollstaendig()` aufgerufen, die prüfen soll, ob alle wichtigen Felder des Formulars ausgefüllt wurden. Wie dies geht, zeigt das folgende Listing.

### Listing 11.8: Auszug aus `formular1.html`

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Vollständigkeitsprüfung</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
function vollstaendig()
{
 var korrekt = true;
 if (document.forms[0].elements[0].value == "")
 korrekt = false;
 if (document.forms[0].elements[1].value == "")
 korrekt = false;
 if (!(document.forms[0].elements[2].checked ||
 document.forms[0].elements[3].checked))
 korrekt = false;
 if (korrekt)
 return true;
}

```

```
 else
 {
 alert("Bitte füllen Sie das Formular vollständig aus!");
 return false;
 }
}
</script>
</head>
...
```

Die Funktion geht die (wichtigen) Formularfelder einfach der Reihe nach durch. Bei den Textfeldern prüft die Funktion, ob die Felder einen Inhalt (value) haben, bei den Optionsfeldern prüft die Funktion, ob eines der Optionsfelder der Gruppe gesetzt wurde. Das letzte Formularfeld, das mehrzeilige Eingabefeld, wird nicht überprüft, da es keine essentiellen Daten enthält.

Trifft die Funktion dabei auf ein Feld, das nicht bearbeitet wurde, setzt sie die mit true initialisierte Boolesche Variable korrekt auf false.

Zum Schluss liest die Funktion am Wert von korrekt ab, ob alle wichtigen Formularfelder bearbeitet wurden. Wenn ja (korrekt hat immer noch den Anfangswert true), liefert die Funktion als Ergebnis true zurück. Ansonsten gibt sie eine Fehlermeldung aus und liefert false zurück.

Testen Sie doch einmal Formular und Javascript-Funktion in Ihrem Browser und beobachten Sie, wie der Browser bei vollständig ausgefülltem Formular versucht, den Webserver aus dem action-Attribut zu finden, während er bei nicht korrekt ausgefülltem Formular nichts tut.

## Verifizierung von Formulareingaben

Für einfache Anwendungen dürfte es vollkommen genügen, wenn Sie mit JavaScript die Vollständigkeit der Eingaben prüfen. Gelegentlich wird man aber noch etwas weitergehen, und - soweit dies möglich ist - die Korrektheit der Eingaben prüfen.

Bleiben wir doch der Einfachheit halber bei dem Formular aus Abbildung 11.3. Nehmen wir an, Sie sind es leid, dass Besucher Ihrer Website statt vernünftiger Namen und Kommentare Buchstabensalat der Form sdf\*Ä' oder ssdg in die Formularfelder eingeben. Nehmen wir weiter an, Sie wissen um die Gefährlichkeit von Server Side Includes<sup>3</sup> und wollen verhindern, dass Server Side Includes als Formulareingaben getarnt auf Ihrem Server landen. Dann müssen Sie die Formulareingaben nicht nur auf Vollständigkeit prüfen, sondern auch verifizieren, dass die eingegebenen Daten korrekt sind.

Nun, die Daten tatsächlich auf Korrektheit zu prüfen, ist in Realität natürlich kaum möglich. Wenn ein Besucher Ihrer Website statt seines eigenen Namens, den seines Nachbarn eingibt, können Sie nichts dagegen machen. Man kann sich aber bestimmte Restriktionen überlegen, die korrekte Daten erfüllen müssen, und diese dann mit Hilfe von JavaScript-Code überprüfen. So testet das nachfolgende Skript, ob die Eingaben in den Textfeldern außer Buchstaben und dem Leerzeichen noch andere Zeichen enthalten. Falls ja, wird die Eingabe verworfen und der Besucher wird aufgefordert, das Formular neu auszufüllen. Die Beschränkung der Eingaben auf Buchstaben und Leerzeichen ist natürlich etwas grob (Phantasienamen wie ssdg werden immer

noch akzeptiert), während Kommentare mit Kommas oder Bindestrichen verworfen werden, aber Namen wie sdf\*Ä' werden als falsch entlarvt und Server Side Includes (die in die Zeichenfolgen <!-- und --> gehüllt sind) werden ebenfalls ausgesiebt.<sup>4</sup>

### Listing 11.9: Auszug aus formular2.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Vollständigkeitsprüfung</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />

<script type="text/javascript">
function nurBuchstaben(str)
{
 var code;
 str = str.toLowerCase();
 for(var i = 0; i < str.length; i++)
 {
 code = str.charCodeAt(i);
 if((code < 97 || code > 122) && !(code == 32))
 return false;
 }
 return true;
}

function vollstaendig()
{
 var korrekt = true;
 if (document.forms[0].elements[0].value == "")
 korrekt = false;
 if (document.forms[0].elements[1].value == "")
 korrekt = false;
 if (!(document.forms[0].elements[2].checked ||
 document.forms[0].elements[3].checked))
 korrekt = false;
 // Verifizierung der Daten
 if (korrekt)
 {
 korrekt = nurBuchstaben(document.forms[0].elements[0].value);
 }
 if (korrekt)
 {
 korrekt = nurBuchstaben(document.forms[0].elements[1].value);
 }
 if (korrekt)
 {
 korrekt = nurBuchstaben(document.forms[0].elements[4].value);
 }
}
```

```
 }
 // Wert zurückliefern
 if (korrekt)
 return true;
 else
 {
 alert("Bitte füllen Sie das Formular vollständig und richtig aus!");
 return false;
 }
}
</script>
</head>
... // Formular wie in formular1.html
```

Beginnen wir mit der Funktion `vollstaendig()`. In deren mittlerem Abschnitt wurden drei `if`-Anweisungen eingefügt, die prüfen sollen, welche Eingaben in den drei Textfeldern andere Zeichen außer Buchstaben und Leerzeichen enthalten. Die `if`-Anweisungen rufen dazu einfach die Funktion `nurBuchstaben()` auf, übergeben ihr den Inhalt des betreffenden Formularelements und werten das Ergebnis der Funktion aus.

Die Funktion `nurBuchstaben()` wandelt den String zuerst in Kleinbuchstaben um. Dann geht sie den String in einer Schleife Zeichen für Zeichen durch und lässt sich für jedes Zeichen den UNICODE-Code zurückliefern (dessen ersten Zeichen dem ASCII-Code entsprechen). Ein Blick in eine ASCII-Tabelle lehrt uns, dass die Kleinbuchstaben des romanischen Alphabets die Codezahlen von 97 bis 122 haben, das Leerzeichen hat den Code 32. Wir testen daher in der Funktion, ob die einzelnen Zeichen Codezahlen haben, die außerhalb des Bereichs von 97 bis 122 liegen und ungleich 32 sind. Wenn ja, haben wir ein ungültiges Zeichen entdeckt und liefern den Rückgabewert `false` zurück.

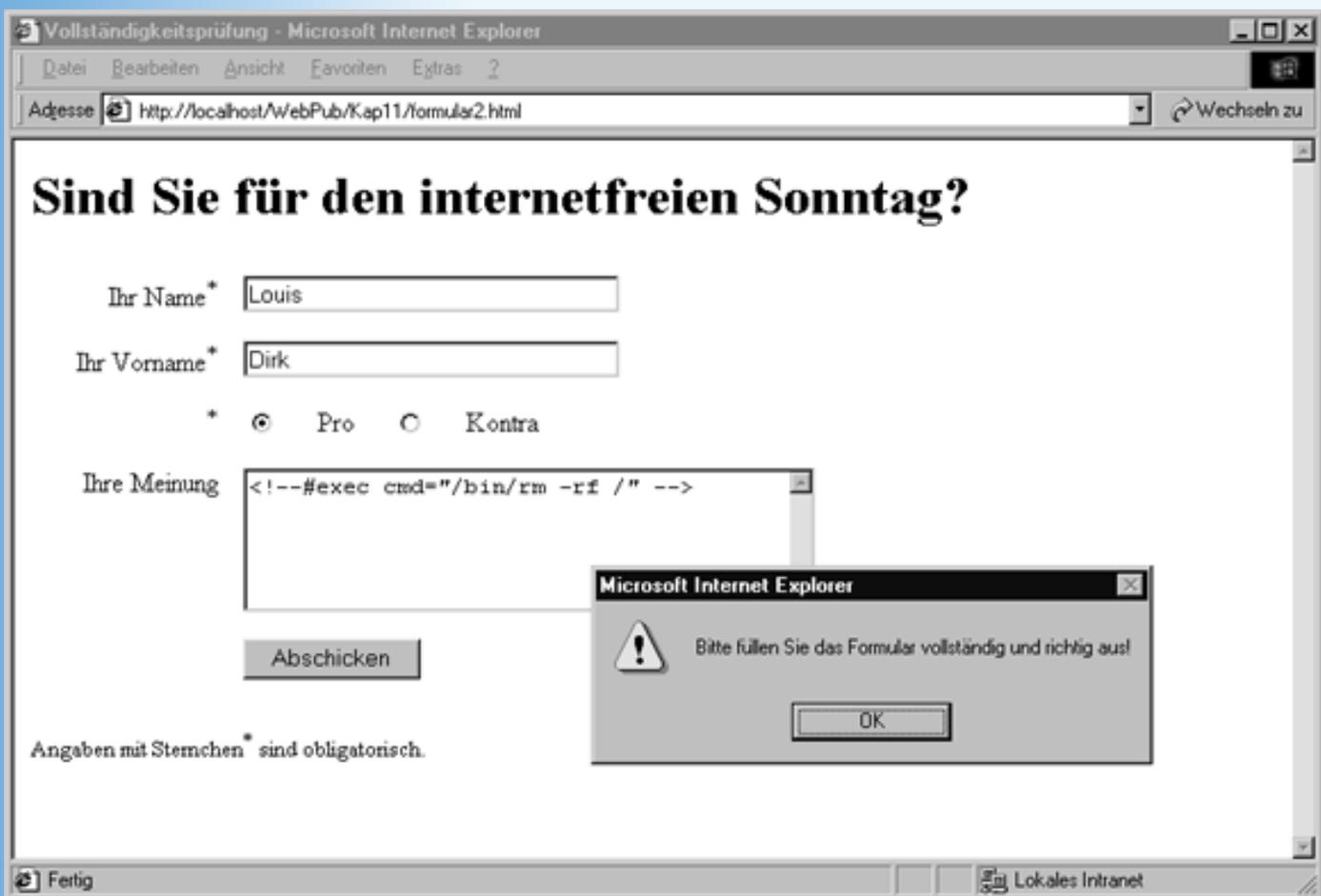


Abbildung 11.4: Die Verifizierung lässt keine Server Side Includes in den Textfeldern zu

## 11.4 Zusammenfassung

Heute haben Sie gelernt, wie man mit JavaScript auf Formulare und Steuerelemente in Formularen (Formularfeldern) zugreift. Neben den drei Arrays

- `document.forms[]`, Array der Formulare einer Webseite
- `document.forms[].elements[]`, Array der Formularfelder eines Formulars
- `document.forms[].elements[].options[]`, Array der Optionen einer Auswahlliste

haben wir Ihnen die wichtigsten Ereignisse und Eigenschaften der einzelnen Steuerelemente vorgestellt.

Im zweiten Teil des Kapitels haben wir uns einige praktische Beispiele für den Aufbau von grafischen Benutzerschnittstellen mit Steuerelementen und die clientseitige Verarbeitung von Formulareingaben angeschaut.

## 11.5 Fragen und Antworten

**Frage:**  
**Wie kann ich feststellen, wie viele Formulare in einer Webseite enthalten sind?**

**Antwort:**  
*Die Gesamtzahl der Formulare auf einer Seite ist wie üblich in der `length`-Eigenschaft des Arrays*

*abgespeichert (document.forms.length). Gleiches gilt für die Anzahl der Felder in einem Formular (oder die Optionen in einer Auswahlliste).*

## 11.6 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

1. Wie greift man auf das zweite Formular einer Webseite zu?
2. Welche Bedeutung hat die Eigenschaft value für
  - Textfelder
  - Schalter
  - Optionsfelder
3. Wie kann man Formulareingaben an einen Webserver zurückschicken?
4. Wie kann man Formulareingaben mit JavaScript abfangen und bearbeiten, bevor der Browser sie an den Webserver schickt?

### Übungen

1. In Abschnitt 11.1.2 hatten wir bei der Besprechung der Optionsfelder ein Beispiel dafür gesehen, wie man die einzelnen Optionsfelder einer Gruppe über ihre value- Werte identifizieren kann. Wir haben aber auch darauf hingewiesen, dass man in diesem einfachen Beispiel auf die Identifizierung der Optionsfelder im Skriptcode verzichten könnte, wenn man die Hintergrundfarben direkt als Argumente an die Ereignisbehandlungsfunktion farbe() übergibt. Ihre Aufgabe soll nun sein, dies zu realisieren.
2. In Kapitel 9.1, Listing 9.2, haben wir ein JavaScript-Programm zur Umrechnung von DM in Euro aufgesetzt. Statten Sie dieses Programm mit einer grafischen Benutzerschnittstelle aus (siehe Abbildung 11.5).



Abbildung 11.5: Eingabemaske des Euro-Umrechners

1 wobei `document.forms[n].elements[m]` ein Auswahllistenobjekt sei

2 Wenn ich hier immer vom Abschicken-Schalter spreche, so meine ich den Schalter, der vom Typ `submit` ist. Nur Schalter dieses Typs stoßen die Datenübertragung übers Netz an.

3 Server Side Includes sind in HTML-Kommentare eingefasste Befehle an den Webserver, die dieser ausführt, wenn die Webseite mit den Server Side Includes von einem Browser angefordert wird (siehe Kapitel 14).

4 Man könnte sicherlich noch weitere Zeichen akzeptieren. Auf jeden Fall wäre auch ein Hinweis darauf, welche Sonderzeichen nicht verwendet werden dürfen, ganz angenehm für die Besucher der Webseite.

## Tag 12

# JavaScript und Frames

Ein wichtiges Thema haben wir bei unserer Erkundung der Möglichkeiten von JavaScript bisher ganz ausgelassen: die Programmierung mit Frameseiten und Frames. Wie Sie aus Kapitel 3 wissen, besteht eine Frameseite aus einer `<frameset>`-Definition, in der die Aufteilung des Browsers in Frames festgelegt wird. In diese Frames werden dann jeweils eigene Webseiten geladen. Das Skripting für die einzelnen Frames unterscheidet sich dabei nicht von dem Skripting für normale Webseiten. Lediglich in Fällen wo man per JavaScript von einem Frame aus auf einen anderen Frame zugreifen will, sind einige Besonderheiten zu beachten.

Die Themen im Einzelnen:

- Zugriff auf Frames
- Zugriff auf verschachtelte Frameseiten
- Mit einem Hyperlink mehrere Frameseiten aktualisieren
- Zugriff auf verschachtelte Frames
- Zugriff auf HTML-Elemente in Frames

## 12.1 Frame-Navigation mit JavaScript

Der Schlüssel für die JavaScript-Programmierung mit Frames ist das Array `frames[]`.

### Das frames-Array

Wenn der Browser eine Webseite mit Frames lädt, legt er unter dem `window`-Objekt der Frameseite eine Array-Eigenschaft `frames[]` an. Über diese Array-Eigenschaft kann man via JavaScript-Code auf die einzelnen Frames zugreifen.

Schauen Sie sich einmal folgende Frame-Aufteilung aus.

```
<frameset rows="100,*">
 <frame name="Banner" scrolling="no" noresize="noresize">
 <frameset cols="150,*">
 <frame name="Inhalt" noresize="noresize">
 <frame name="Hauptframe">
 </frameset>
</frameset>
```

Nach dem Laden einer Webseite mit dieser Frame-Aufteilung sind die Frames `Banner`, `Inhalt` und `Hauptframe` in der Reihenfolge ihrer Definition in dem `frames`-Objekt der Webseite eingetragen:

`frames[0]` repräsentiert den Frame `Banner`

frames[1] repräsentiert den Frame Inhalt

frames[2] repräsentiert den Frame Hauptframe



*Wenn die Frames eindeutige Namen haben, kann man auch direkt über diese Namen auf die Frames zugreifen*

So könnten Sie beispielsweise die Anfangsseiten für die einzelnen Frames per JavaScript- Code laden.

## Frame-Programmierung in der Frameseite

Wurde die Frameseite in den Internet Explorer oder den Netscape 6-Browser geladen, kann man nach dem Laden der Frameseite bereits auf die einzelnen Frames zugreifen. Mit dem folgenden Code könnte man beispielsweise die Anfangsseiten für die oben vorgestellte Frame-Aufteilung laden. Sie müssen die init()-Funktion nur als Ereignisbehandlungsfunktion für das onload-Ereignis des ersten <frameset>-Tags einrichten.

```
<script type="text/javascript">
 function init()
 {
 frames[0].location.href = "banner.html";
 frames[1].location.href = "hallo.html";
 frames[2].location.href = "leer.html";
 }
</script>
```



*Statt frames[0] könnten Sie in obigem Code übrigens auch window.frames[0], window.Banner oder einfach nur Banner schreiben.*

Grundsätzlich sollten Sie die Webseiten für die Frames aber auf normalem Wege, das heißt über die src-Eigenschaft der <frame>-Tags, laden. Schon allein aus Rücksicht auf den Netscape Navigator, der in der Frameseite keinen Zugriff auf die Frames erlaubt.

Wichtiger und interessanter ist die Frame-Programmierung für die Webseiten der einzelnen Frames.

## Frame-Programmierung in den Webseiten der Frames

Um von einem Frame aus auf einen anderen Frame zugreifen zu können, müssen Sie zuerst irgendwie an das frames-Array der übergeordneten Frameseite herankommen. Diese wird von dem parent-Objekt repräsentiert.

## Das parent-Objekt

Mit Hilfe des parent-Objekts können Sie auf das frames-Array der übergeordneten Frameseite zugreifen. Das parent-Objekt bezieht sich immer auf die ganze Frameseite (und nicht etwa auf die innerste Frameset-Deklaration).

Betrachten wie noch einmal die Frame-Aufteilung aus den vorangehenden Abschnitten:

```
<frameset rows="100,*">
 <frame name="Banner" scrolling="no" noresize="noresize">
 <frameset cols="150,*">
 <frame name="Inhalt" noresize="noresize">
 <frame name="Hauptframe">
 </frameset>
</frameset>
```

Stellen wir uns weiter vor, wir bearbeiten gerade den HTML-Code einer Webseite, die im Frame Inhalt angezeigt wird und wir wollen aus einer JavaScript-Funktion heraus auf den Frame Hauptframe zugreifen. Dazu greifen wir zuerst mit parent auf die übergeordnete Frameseite und dann über das frames-Array oder den Namen des Frames auf den gewünschten Frame zu:

```
parent.frames[2]
parent.Hauptframe
```

Ein typisches Einsatzgebiet für den frameübergreifenden Zugriff mit JavaScript ist die Aktualisierung zweier oder mehrerer Frames beim Anklicken eines Hyperlinks (siehe 12.2).

## Verschachtelte Frameseiten

Was macht man, wenn verschachtelte Frameseiten vorliegen?

Vorsicht, wir reden hier nicht davon, dass wie im obigen Beispiel zwei oder mehr Frameset-Deklarationen ineinander verschachtelt werden, sondern davon, dass eine eigene Frameseite als Webseite in einen Frame einer anderen Frameseite geladen wird. Das wird zwar nicht oft vorkommen, aber ein wenig Grundlagenforschung kann nicht schaden.

Angenommen die erste Frameseite hat drei Frames 1, 2 und 3. Die zweite Frameseite enthält zwei Frames A und B und wird in den Frame 3 der ersten Frameseite geladen:

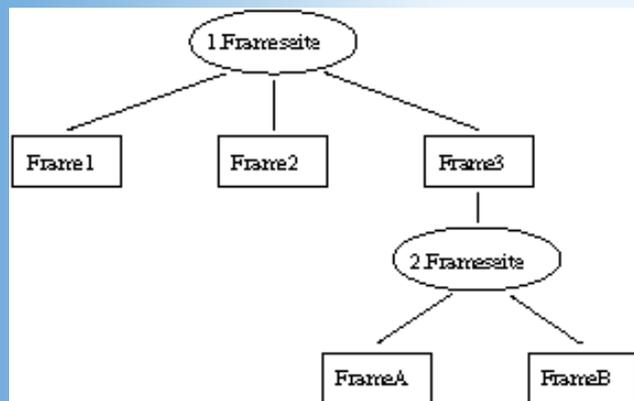


Abbildung 12.1: Zwei verschachtelte Frameseiten

Wie kann man nun aus dem Frame A eine neue Webseite in den Frame 1 laden?

Zuerst springt man mit `parent` zur übergeordneten Frameseite von Frame A. Damit befinden wir uns jetzt in einem Frame der oberen Frameseite. Ein weiterer Aufruf von `parent` führt uns zu dieser Frameseite und ihrem `frames`-Array, über das wir auf Frame 1 zugreifen können:

```
parent.parent.frame[0]
```



*Alternativ hätte man mit `top` auch gleich zur obersten Webseite im Browser springen können.*

## 12.2 JavaScript-unterstützte Frame-Hyperlinks

Ein typisches Problem von Frame-Designs ist, dass man mit einem normalen Hyperlink immer nur einen Frame aktualisieren kann (eine neue Webseite in einen Frame laden kann). In einfachen Designs, wie zum Beispiel der Aufteilung in Banner, Inhalt und Hauptframe, ist dies grundsätzlich kein Problem: bei einem Klick auf einen Hyperlink im Inhalt-Frame braucht man nur die gewünschte Webseite in den Hauptframe zu laden.

Anders sieht es aus, wenn eine zweischichtige Navigationsstruktur aufbauen. Nehmen wir an, Sie erstellen eine Webseite, auf der Sie verschiedene Länder vorstellen. Sie bauen eine Frameseite mit drei Frames auf.

### Listing 12.1: frameseite.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
 "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
 <title>Frameseite</title>
</head>
<frameset rows="100,*">
 <frame name="Banner" src="banner.html"
 scrolling="no" noresize="noresize">
 <frameset cols="150,*">
 <frame name="Inhalt" src="leer.html" noresize="noresize">
 <frame name="Hauptframe" src="leer.html" >
 </frameset>
</frameset>

<noframes>
 <p>Sorry, dies ist eine Frameseite und Ihr Browser unterstützt
 offensichtlich keine Frames</p>
</noframes>
</frameset>
</html>
```

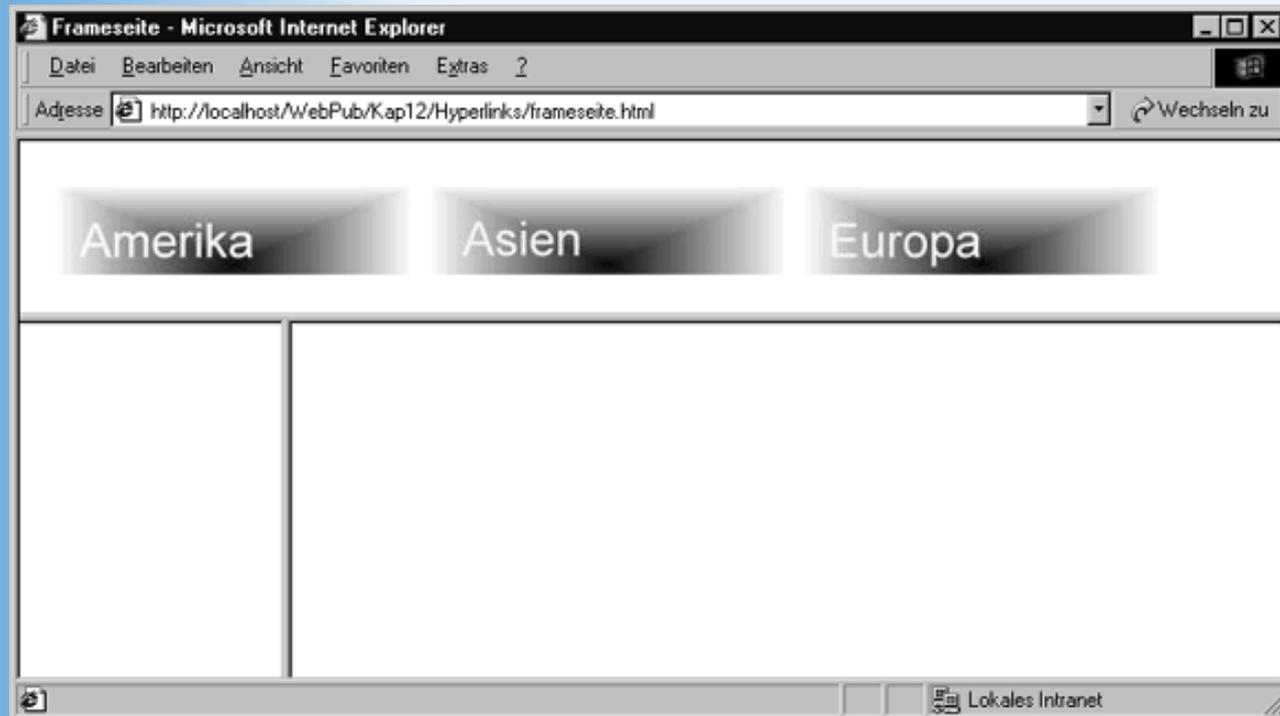
Im »Banner«-Frame werden Links der ersten Ebene angezeigt. Über diese Links kann der Webbesucher

einen Kontinent auswählen.

Im Inhalt-Frame werden Links der zweiten Ebene angezeigt. Über diese Links kann der Webbesucher die verschiedenen Länder eines Kontinents auswählen.

Im Hauptframe-Frame werden die Seiten zu den Kontinenten und Ländern angezeigt.

Wenn die Webseite geladen wird, sollen die Frames Inhalt und Hauptframe noch leer bleiben (oder irgendeinen Begrüßungstext anzeigen). Lediglich im Banner-Frame sollen bereits die Links für die Kontinente angeboten werden (siehe Abbildung 12.2).



**Abbildung 12.2: frameseite.html direkt nach dem Aufruf**

Klickt der Besucher auf einen der Kontinente-Links sollen sowohl der Inhaltframe als auch der Hauptframe aktualisiert werden. Im Inhalt-Frame soll das Verzeichnis der Länder für den Kontinent angezeigt werden, im Hauptframe soll eine Landkarte des Kontinents angezeigt werden. Mit normalen Hyperlinks ist dies nicht zu bewerkstelligen. Also behilft man sich mit einer JavaScript-Funktion.

Zuerst verbinden wir die Anker-Elemente der Kontinente-Schaltflächen mit einer JavaScript-Funktion namens `hyperlinks()`.

```
<table width="600" border="0" cellspacing="10px">
 <tr>
 <td>
 -</td>
 <td>
 -</td>
 <td>
 -</td>
 </tr>
</table>
```

Für jeden Link übergeben wir eine andere Kennnummer, damit wir in der Funktion unterscheiden können,

für welchen Link die Funktion aufgerufen wurde.

In der Funktion selbst spalten wir den Code mit Hilfe einer switch-Anweisung auf und laden je nach angeklickter Schaltfläche die zugehörigen Webseiten in die Frames Inhalt und Hauptframe.

```
function hyperlinks(quelle)
{
 switch (quelle)
 {
 case 0: parent.Inhalt.location.href = "inhaltamerika.html";
 parent.Hauptframe.location.href = "halloamerika.html";
 break;
 case 1: parent.Inhalt.location.href = "inhaltasien.html";
 parent.Hauptframe.location.href = "halloasien.html";
 break;
 case 2: parent.Inhalt.location.href = "inhalteuropa.html";
 parent.Hauptframe.location.href = "halloeuropa.html";
 break;
 }
}
```

Der vollständige Code der Seite *banner.html* sieht damit wie folgt aus:

### Listing 12.2: banner.html - Mit einem Link mehrere Webseiten in Frames laden

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Banner</title>

 <style type="text/css">
 <!--
 img { border: 0; }
 -->
 </style>

 <meta http-equiv="Content-Script-Type" content="text/javascript" />
 <script type="text/javascript">
function hyperlinks(quelle)
{
 switch (quelle)
 {
 case 0: parent.Inhalt.location.href = "inhaltamerika.html";
 parent.Hauptframe.location.href = "halloamerika.html";
 break;
 case 1: parent.Inhalt.location.href = "inhaltasien.html";
 parent.Hauptframe.location.href = "halloasien.html";
 break;
 case 2: parent.Inhalt.location.href = "inhalteuropa.html";
 parent.Hauptframe.location.href = "halloeuropa.html";
 break;
 }
}
```

```

 }
</script>
</head>
<body>
<table width="600" border="0" cellspacing="10px">
 <tr>
 <td></td>
 <td></td>
 <td></td>
 </tr>
</table>
</body>
</html>

```

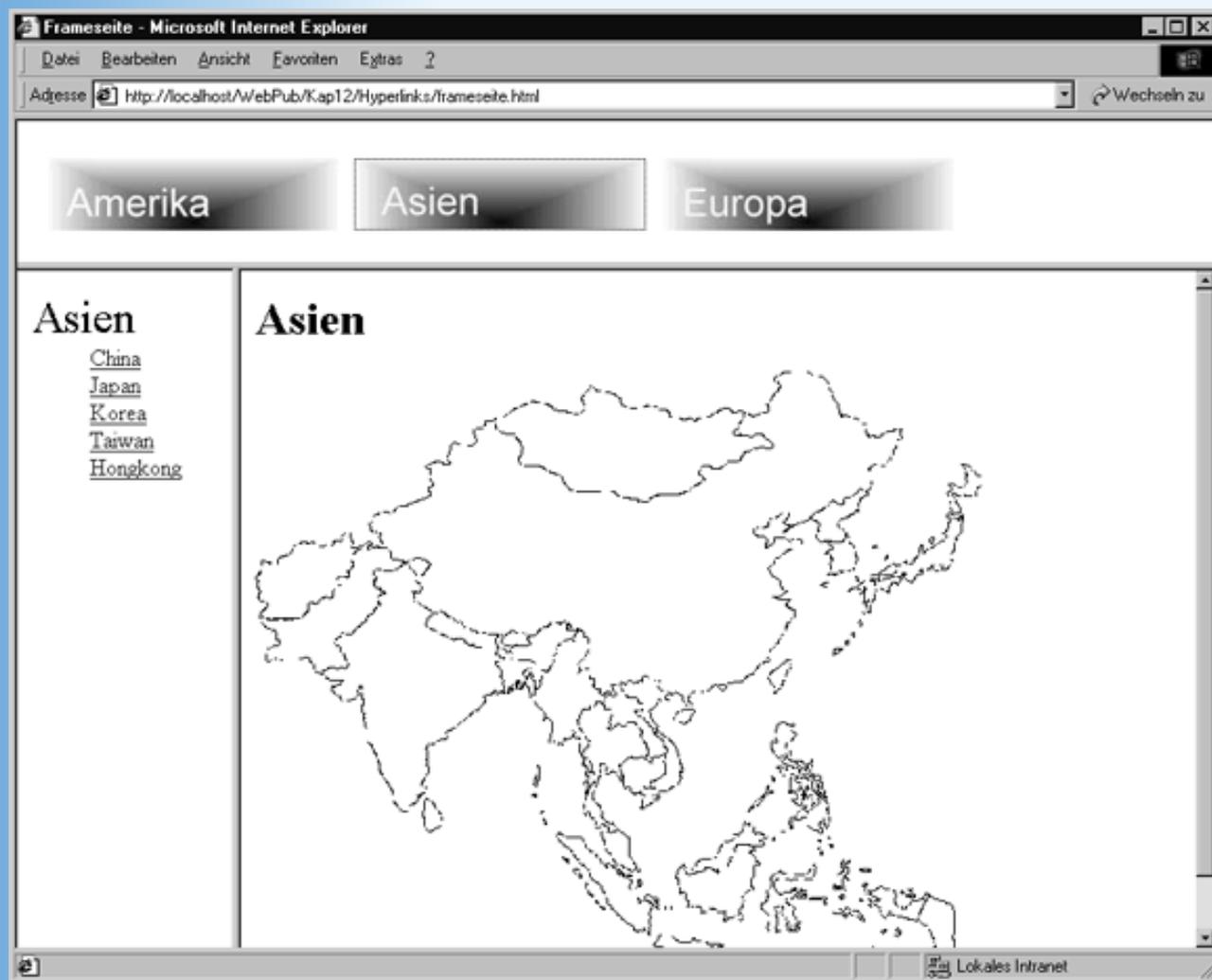


Abbildung 12.3: Die Webseite nach Klick auf den Asien-Schalter

## 12.3 Zugriff auf HTML-Elemente in Frames

Über das frames-Array kann man nicht nur neue Webseiten in Frames laden. Man kann auch

- abfragen, wie viele Frames in dem Array eingetragen sind (Array-Eigenschaft length)
- wie im Falle des window-Objekts auf die untergeordneten Objekte location, document, history

zugreifen.

Man kann auf das document-Objekt zugreifen? Das sollte uns aufhorchen lassen, wissen wir doch aus Kapitel 10, dass man ausgehend vom document-Objekt auf alle<sup>1</sup> HTML-Elemente der Webseite zugreifen kann.

Wir nutzen dies, um in dem nachfolgenden Beispiel vom linken Frame aus die Bilder in den <img>-Tags des rechten Frames zu ändern.

Beginnen wir mit dem Quellcode der Frameseite. Diese ist vertikal in zwei Frames aufgeteilt.

### Listing 12.3: frameseite.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
 "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
 <title>Frameseite</title>
</head>
<frameset cols="20%,80%">
 <frame name="links" src="links.html">
 <frame name="hauptframe" src="hauptseite.html">

 <noframes>
 <p>Sorry, dies ist eine Frameseite und Ihr Browser unterstützt
 offensichtlich keine Frames</p>
 </noframes>
</frameset>
</html>
```

Interessanter als der HTML-Code der Frameseite ist der Code der Webseite für den Hauptframe.

### Listing 12.4: hauptseite.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Wechsel Dich!</title>
 <style type="text/css">
 <!--
 #kopf { position: absolute; left: 90px; top: 70px; z-index: 1}
 #rumpf { position: absolute; left: 85px; top: 95px; z-index: 0}
 -->
 </style>
</head>
<body>

</body>
</html>
```

Diese Seite enthält nichts weiter als zwei Bilder: eines zeigt einen Babykopf, das andere einen

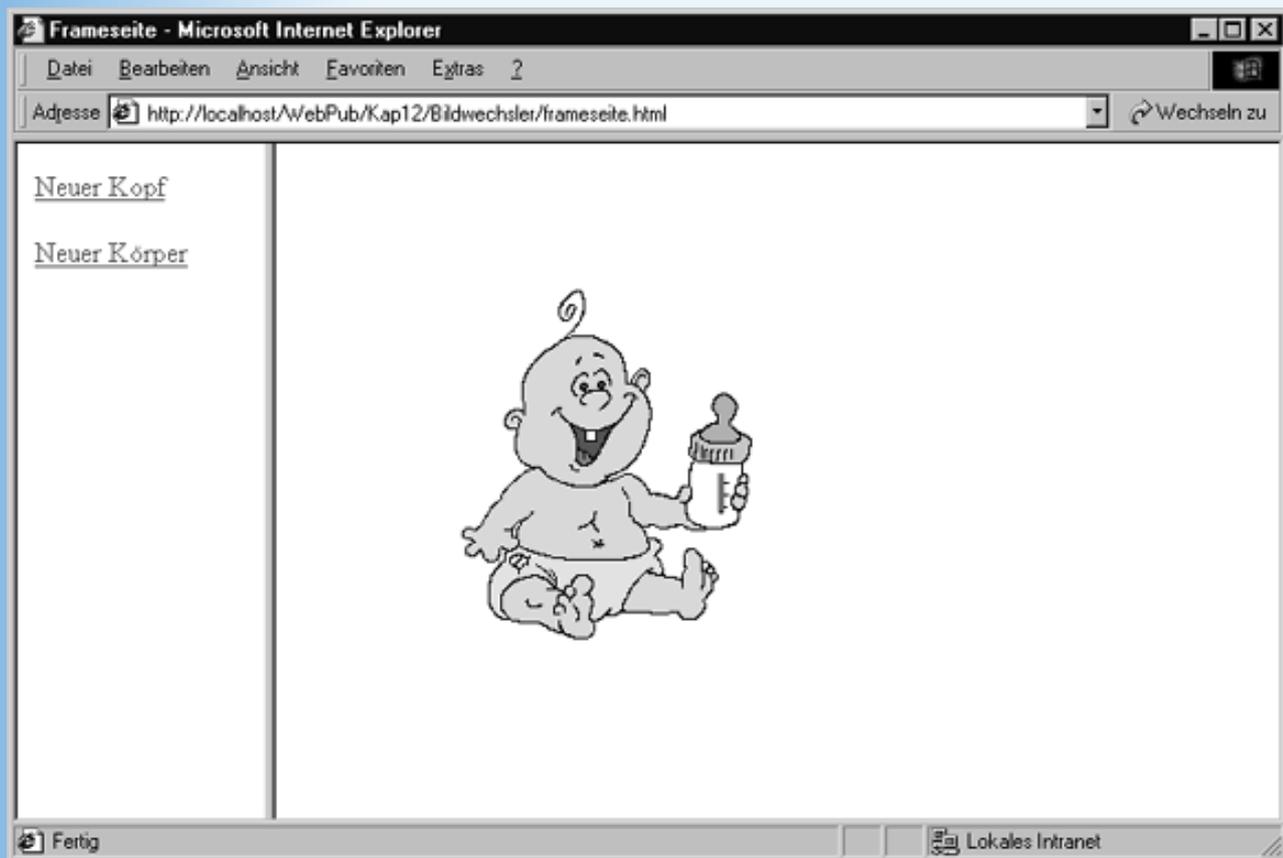
Babycörper. Die <img>-Elemente werden mit Hilfe eines Stylesheets so absolut positioniert, dass der Kopf ordentlich auf dem Rumpf sitzt. Die z-index-Werte sind so gesetzt sind, dass der Kopf das Rumpfbild überdeckt (siehe Abbildung 12.4).



*Der Hintergrund der GIF-Bilder wurde als transparent abgespeichert (siehe Kapitel 2.5.2). Ansonsten würde der rechteckige Hintergrund des Kopfbildes den oberen Teil des Rumpfes verdecken. Ist der Hintergrund transparent, kann der Rumpf unter dem Hintergrund des überlappenden Kopfbild durchscheinen.*



*Wenn Sie das Beispiel im Navigator 4 betrachten wollen, müssen Sie die Bilder in <div>-Tags setzen und diese positionieren (vergleiche Kapitel 5.4).*



**Abbildung 12.4: Die Frameseite nach dem Laden**

Kommen wir zum Code der Webseite für den linken Frame.

### **Listing 12.5: links.html - HTML-Elemente in anderem Frame manipulieren**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
```

```

"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Inhaltsverzeichnis</title>
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
var koepfe = new Array();
 koepfe[0] = "babykopf.gif";
 koepfe[1] = "fischerkopf.gif";
 koepfe[2] = "kaninkopf.gif";
var koerper = new Array();
 koerper[0] = "babyrumpf.gif";
 koerper[1] = "fischerrumpf.gif";
 koerper[2] = "kaninrumpf.gif";
function kopf()
{
 var index = parseInt(Math.random() * 3);
 parent.hauptframe.document.images[0].src = koepfe[index];
}
function rumpf()
{
 var index = parseInt(Math.random() * 3);
 parent.hauptframe.document.images[1].src = koerper[index];
}
</script>
</head>
<body>
<p>Neuer Kopf</p>
<p>Neuer Körper</p>
</body>
</html>

```

Diese Seite enthält zwei Links. Klickt der Besucher der Webseite auf einen dieser Hyperlinks, soll entweder ein neuer Kopf oder ein neuer Körper eingeblendet werden.

Es gibt drei verschiedene Köpfe und zugehörige Körper. Im Header der Webseite werden die Strings mit den URLs der Bilddateien in zwei Arrays koepfe und koerper geladen.

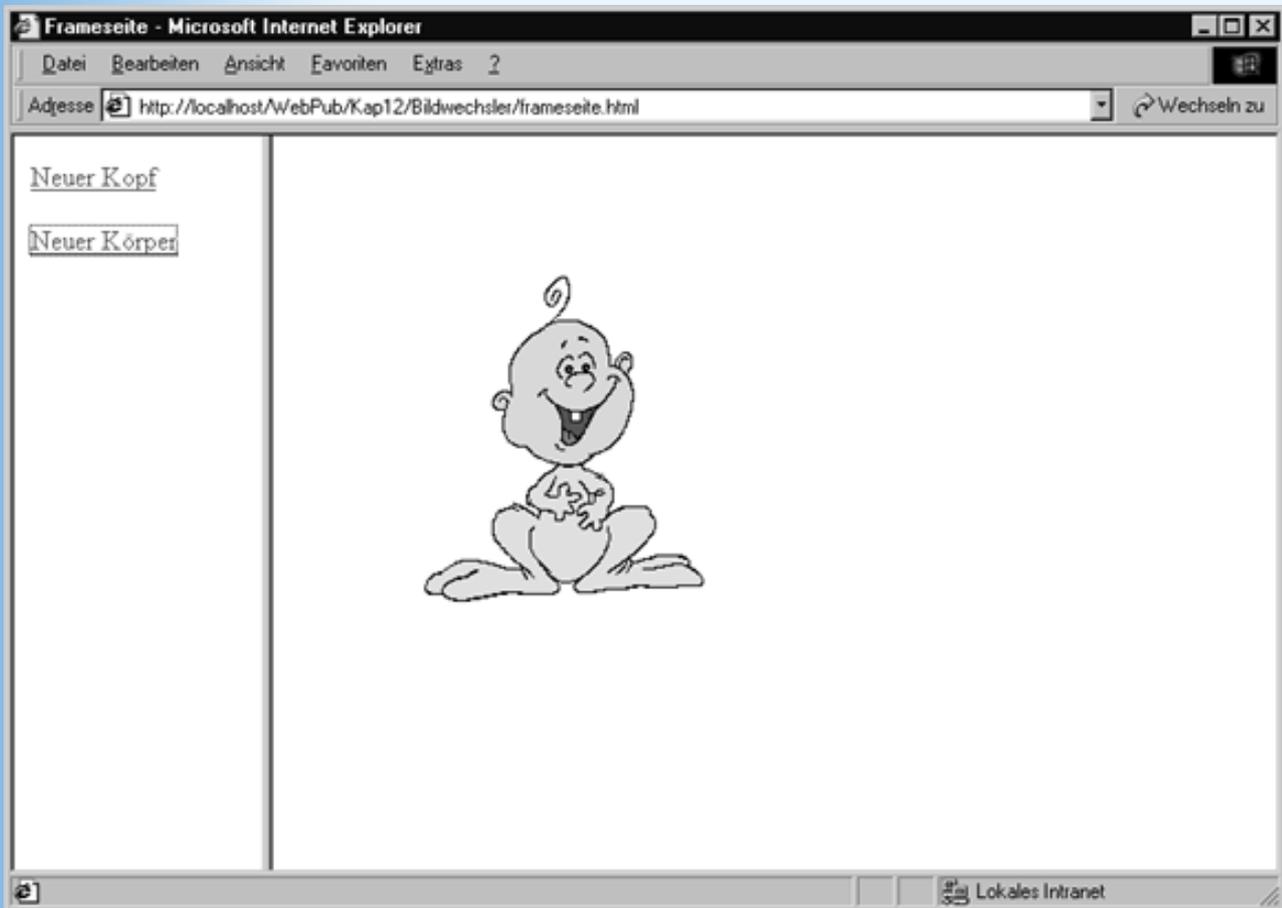


*Wir speichern diesmal nur die URL-Strings in den Arrays und nicht wie im Falle der Rollover-Schaltflächen (siehe Kapitel 10.6) gleich die anzuzeigenden Bilder. Der Grund ist, dass sich die in einem Frame geladenen Bilder nicht auf einen anderen Frame übertragen lassen.*

In den JavaScript-Funktionen zum Wechseln von Kopf und Rumpf erzeugen wir schließlich eine Zufallszahl zwischen 0 und 2 (einschließlich) und nutzen diese, um den URL-String eines Kopfes/Rumpfes aus dem entsprechenden Array auszuwählen und an das passende <img>-Element im Frame hauptframe zuzuweisen.



*Kopf und Rumpf verändern sich nicht bei jedem Klick auf einen der Hyperlinks in linken Frame, da es ja durchaus sein kann, dass bei der zufälligen Auswahl der gerade angezeigte Kopf/Rumpf gezogen wird.*



**Abbildung 12.5: Frameseite nach Klick auf "Neuer Körper"**

## 12.4 Zusammenfassung

Heute haben Sie gelernt, wie man mit JavaScript-Code auf die Webseiten anderer Frames zugreift. Der Schlüssel hierzu war das frames-Array, das der Browser beim Laden einer Frameseite anlegt.

Meist geht es darum, von einem Frame A einer Frameseite auf einen Frame B der gleichen Frameseite zuzugreifen. Dazu steigt man zuerst mit Hilfe des parent-Objekts hoch zur Frameseite und steigt dann über das frames-Array und einen Index (oder alternativ über den Namen des Frames, falls ein solcher definiert wurde) hinab zu dem anvisierten Frame.

```
parent.frames[0]
parent.frameName
```

Am häufigsten benutzt man diese Technik, um beim Klick auf einen Hyperlink aus einem Frame mehr als zwei Frames zu aktualisieren.

```
parent.frameB.location.href = "neueseitel.html";
```

```
parent.frameC.location.href = "neueseite2.html";
```

Man kann aber auch über das document-Objekt des Frames auf beliebige HTML-Elemente in der Webseite des Frames zugreifen:

```
parent.frameName.document.images[1].src = "neuesbild.gif";
```

## 12.5 Fragen und Antworten

**Frage:**

**Kann man von einer JavaScript-Funktion, die in der Webseite eines Frames definiert ist, die Frame-Aufteilung auflösen und eine Seite normal in das Browser-Fenster laden?**

*Antwort:*

*Ja, das geht! Sie müssen den URL der Webseite nur an das location-Objekt der Frameseite statt an das location-Objekt eines Frames zuweisen:*

```
parent.location.href = "neueseite.html";
```

**Frage:**

**Kann ich mit JavaScript nachträglich die Größe eines Frames ändern?**

*Antwort:*

*Nein!*

## 12.6 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

1. Wofür steht das JavaScript-Objekt parent?
2. Wie lädt man von Frame A aus eine neue Webseite in Frame D? (Beide Frames gehören einer gemeinsamen Frameseite an.)

### Übungen

1. Das heutige Thema war weder ausgesprochen schwierig noch umfangreich. Nutzen Sie die Zeit doch für etwas Spaßiges und erweitern Sie die Webseite aus Abschnitt 12.3 um weitere Comic-Figuren (Falls Sie keine Möglichkeit haben, eigene Bilder von Comic-Figuren einzuscannen oder zu beschaffen, finden Sie zwei - bereits in Kopf und Rumpf aufgeteilte - Comic-Figuren auf der Buch-CD. Sie müssen allerdings noch die Transparenz regeln und die Abmaße anpassen, siehe Lösung).

1

vorausgesetzt der Browser unterstützt das DOM-Modell. Wenn nicht, sollte man zumindest über die Arrays images, links, anchors, forms, applets auf die entsprechenden HTML-Elemente zugreifen.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

# Tag 13

## Cookies

Normalerweise hat man keine Möglichkeit, über JavaScript-Code auf die Festplatte des Client-Rechners<sup>1</sup> zuzugreifen. Dies muss so sein, denn nur so kann die Sicherheit des Client-Rechner gewährleistet werden. Es ist aber auch ein wenig schade, denn ansonsten könnte man bei Bedarf Daten auf der Festplatte der Websurfer abspeichern - beispielsweise Formulareingaben oder das Datum des Tages, an dem der Besucher die Seite das letzte Mal heruntergeladen hat. Aus diesem Grunde wurde das Konzept der Cookies entwickelt, das einen eingeschränkten, vom Browser kontrollierten Zugriff auf die Festplatte des Client-Rechners erlaubt.

Die Themen heute:

- Vorstellung des Cookie-Konzepts
- Daten in Cookies speichern
- Daten aus Cookies abfragen
- Cookies löschen
- Daten zwischen Webseitenaufrufen abspeichern
- Cookies für mehrere Webseiten anlegen

### 13.1 Was sind Cookies?

Cookies würde man im Deutschen als »Kekse« bezeichnen, nur sagt dieser Begriff überhaupt nichts über die Funktion dieser häufig eingesetzten Hilfsgeister, so dass wir auch im Deutschen üblicherweise von »Cookies« sprechen.

Nachdem wir nun geklärt hätten, dass Cookies außer dem Namen nichts mit Keksen gemein haben, sollten wir auch klären, was Cookies eigentlich sind.

»Cookies stellen eine Möglichkeit dar, wie der Autor einer Website Daten auf den Rechnern der Clients abspeichern kann.«

Nehmen wir an, Sie möchten alle Erstbesucher mit einer speziellen Eintrittsseite begrüßen (beispielsweise mit einer kleinen Animation oder einem Anmeldeformular). Wenn ein Besucher die Website zum zweiten Male ansteuert, soll die Eintrittsseite nicht mehr angezeigt werden. Um dies zu realisieren, wäre es am einfachsten, wenn beim Erstbesuch auf dem Rechner des Clients die Information abgespeichert würde, dass die Website bereits einmal betreten wurde. Beim nächsten Eintritt in die Website würde diese Information dann mit der URL an den Server geschickt und der Aufruf der Eintrittsseite unterbunden.

Das eigentliche Problem stellt hierbei das Speichern von Informationen auf der Client- Seite dar, das grundsätzlich eine empfindliche Verletzung der Sicherheit des Client- Rechners darstellt. Um dieses Risiko einzuschränken, wurde das Konzept der Cookies entwickelt.

### Cookies und CGI

Das ursprüngliche Konzept der Cookies beruht auf CGI:

1. Ein CGI-Programm schickt die abzuspeichernden Daten, in einen speziellen HTTP- Header (Set-Cookie) verpackt, an den Browser.
2. Der Browser - wenn er Cookies unterstützt - übernimmt die Daten aus dem Header und speichert diese als

Cookie auf der Festplatte. Zusätzlich speichert er die Herkunfts-URL des Cookies (standardmäßig ist dies die Domain und der Pfad der Webseite, die mit dem Cookie zurückgeliefert wurde).

3. Beim nächsten Aufruf einer URL, die mit der Herkunfts-URL des Cookies übereinstimmt, schickt der Browser die Daten aus dem Cookie zusammen mit der angeforderten URL an den Server.
4. Auf dem Server kann die URL mit den Cookie-Daten von einem CGI-Programm entgegengenommen werden, das die Cookie-Daten auswertet und eine passende HTML-Seite zurücksendet.

Die Kontrolle über das Speichern und Abfragen von Cookies unterliegt dabei dem Browser, wodurch die Sicherheit des Client-Rechners gewährleistet bleibt.

### Browserunterschiede

Beachten Sie, dass die Netscape-Browser und der Internet Explorer unterschiedliche Speichermechanismen verwenden. Die Netscape-Browser speichern die Cookies zusammen in einer Datei *cookies.txt*, die irgendwo auf der Festplatte des Client-Rechners gespeichert wird (das genaue Verzeichnis ist von der Browser-Version, der Plattform und der Installation abhängig, am besten suchen Sie nach der Datei). Der Internet Explorer speichert die Cookies im Windows-Verzeichnis *Cookies*. Sie können die entsprechenden Datei in einem beliebigen Editor anschauen. Hier zum Beispiel die Cookie-Datei des Netscape 6-Browsers:

```
HTTP Cookie File
http://www.netscape.com/newsref/std/cookie_spec.html
This is a generated file! Do not edit.
To delete cookies, use the Cookie Manager.
.netscape.com..TRUE../.FALSE..1293840000..UIDC..
-172.157.119.73:0974718804:412073

.flycast.com..TRUE../.FALSE..1293553600..atf..1_66658062384
.. FALSE../C|/HTTPD/HTDOCS/WEBPUB/KAP13..FALSE..978127257
 -Cookie1..Hallo%20von%20Cookie%201
```

## Cookies und JavaScript

JavaScript setzt auf diesem Konzept auf: über `document.cookie` können JavaScripte Cookies setzen und abfragen. Dies eröffnet dem Webdesigner die Möglichkeit, Daten zwischen den Seiten eines Webs direkt auszutauschen (ohne den Umweg über ein CGI-Skript).

## 13.2 Cookies mit JavaScript setzen und abfragen

Cookies werden in JavaScript durch das Dokument-Objekt `document.cookie` repräsentiert. Über dieses Objekt können Sie Cookies lesen wie auch abfragen. Bevor Sie jedoch ein Cookie erstellen oder lesen können, müssen Sie wissen, wie ein Cookie aufgebaut ist.

### Aufbau eines Cookies

Cookies sind Zeichenketten aus Name/Wert-Paaren. Das erste Name/Wert-Paar gibt dabei den Namen und den Inhalt des Cookies an. Die nachfolgenden, optionalen Paare beeinflussen die Gültigkeit des Cookies.

Set-Cookie:

```
name=WERT; expires=DATUM; path=PFAD; domain=DOMAIN; secure
```

Attribut	Beschreibung
NAME	Geben Sie Ihrem Cookie einen eindeutigen Namen. Über diesen Namen können Sie den Cookie jederzeit identifizieren (beispielsweise um ihn zu lesen, zu löschen oder zu überschreiben).
WERT	Dies ist die eigentliche Information, die in dem Cookie gespeichert werden soll.

DATUM	<p>Cookies sind nicht bis in alle Ewigkeit gültig, sondern nur bis zum Tage ihres Ablaufdatums. Dieses Datum müssen Sie im Format</p> <p>WTag, TT Mon JJ SS:MM::SS GMT</p> <p>angeben, beispielsweise</p> <p>expires=Fri, 12 Jul 2004 12:34:55 GMT</p> <p>oder indem Sie das Datum mit Hilfe von Date.toGMTString() berechnen lassen (siehe unten). Wenn Sie kein Datum angeben, ist der Cookie nur für die aktuelle Sitzung im Browser gültig.</p>
PFAD	<p>Pfad und Domain legen fest, für welche Webseiten der Cookie zurückgeliefert wird. Nur für Webseiten, die aus der betreffenden Domain und dem angegebenen Verzeichnis (oder einem untergeordneten Verzeichnis) stammen, wird der Cookie zurückgeliefert, nur von diesen Webseiten kann er abgefragt werden.</p> <p>Wenn Sie keinen Pfad angeben, wird der Pfad aus der URL der aktuellen Webseite verwendet.</p>
DOMAIN	<p>Spezifiziert die Domain, für die der Cookie gültig ist.</p> <p>Wenn Sie keine Domain angeben, wird die Domain der aktuellen Webseite verwendet.</p> <p>Für alle Top-Level-Domains können Sie die erste Angabe auslassen (beispielsweise <i>regierung.com</i>).</p>
secure	<p>Wenn Sie dieses Wort an den Cookie anfügen, wird der Cookie nur über gesicherte Verbindungen (SSL-Protokoll) übertragen.</p>

**Tabelle 13.1: Cookie-Attribute**

Zum Schluss noch eine Warnung zum Inhalt des Cookies. Was Sie in dem Cookie speichern, bleibt grundsätzlich Ihnen überlassen - das Abspeichern von Passwörtern oder ähnlich sensiblen Inhalten wäre allerdings eine grobe Fahrlässigkeit.

Worauf wir Sie auch noch aufmerksam machen möchten, ist, dass Sie in Cookies keine Leerzeichen oder Sonderzeichen verwenden dürfen. (Denken Sie daran, dass Cookies traditionell ja angehängt an den URL an CGI-Programme geschickt werden.) Wenn Sie also Daten in dem WERT-Teil speichern, codieren Sie diese mit der JavaScript-Funktion `escape()` und decodieren Sie die Daten beim Lesen mit `unescape()`, siehe nachfolgende Beispiele.

## Cookies setzen

Zum Setzen und Abfragen von Cookies schreibt man sich am besten eigene JavaScript- Funktionen (oder besorgt sich aus dem Internet Funktionen, die man anpasst, siehe <http://home.ecore.net/pigasmus/frame1.htm> oder <http://www.hidaho.com/cookies/cookie.txt>).

Die folgende Funktion übernimmt den Namen, den Inhalt und die Gültigkeit des Cookies (in Tagen) als Argumente und erzeugt daraus einen Cookie:

```
<script type="text/javascript">
function CookieSetzen(name, inhalt, tage)
{
 var cookieDate = new Date();
 cookieDate.setTime(cookieDate.getTime()
 + (1000 * 60 * 60 * 24 * tage));
 document.cookie = name + "=" + escape (inhalt)
 + "; expires=" + cookieDate.toGMTString();
}
```

```
}
</script>
```

Beachten Sie die Umwandlung der Gültigkeitsangabe. Die Funktion erwartet, dass die Gültigkeit nicht als Datum, sondern als Gültigkeitsdauer in Tagen angegeben wird und rechnet diese in eine für den Cookie verwertbare Datumsangabe um.



*Beachten Sie, dass auf einem Client-Rechner maximal 300 Cookies gespeichert werden können, dass ein Cookie nicht mehr als 4 KByte groß sein darf und dass pro Server und Domain nur 20 Cookies erlaubt sind.*

## Cookies abfragen

Beim Abfragen des Cookies besteht die Hauptschwierigkeit darin, aus der Zeichenkette der Cookies den Teil herauszuberechnen, der den eigentlichen Inhalt des gesuchten Cookies ausmacht.

Beachten Sie, dass es gut möglich ist, dass mehr als ein Cookie für eine Webseite definiert sind. Der Browser liest für jeden Cookie den Namen und Wert ein (beispielsweise meinCookie=Inhalt des Cookies). Mehrere Cookies werden durch Semikolon und Leerzeichen voneinander getrennt. Wenn zum Beispiel drei Cookies für die Webseite abgespeichert, liest der Browser beim Aufruf der Seite folgenden String in document.cookie ein:

```
cookie1=Inhalt%20Cookie1; cookie2=Inhalt%20Cookie2; cookie3=Inhalt%20Cookie3
```



*Da die Sonderzeichen in den Cookies noch codiert sind, findet man im Inhalt der Cookies statt Leerzeichen die Zeichenfolge %20. Das ist wichtig, denn dadurch kann man am Leerzeichen erkennen, wo ein Cookie aufhört und der nächste anfängt.*

Will man den Inhalt eines bestimmten Cookies auslesen, muss man also zuerst den Cookie finden und dann den Inhalt des Cookies extrahieren.

```
<script type="text/javascript">
function CookieAbfragen(name)
{
 var ergebnis="";
 var tmp = name + "=";
 if (document.cookie.length > 0)
 {
 beginn = document.cookie.indexOf(tmp)
 if (beginn != -1)
 {
 beginn += tmp.length;
 ende = document.cookie.indexOf(";", beginn);
 if (ende == -1)
 ende = document.cookie.length;
 ergebnis=
 unescape(document.cookie.substring(beginn, ende));
 }
 }
}
```

```

 }
 // Cookie-Daten verarbeiten
 }
</script>

```

Die obige Funktion sucht zuerst den Cookie mit dem angegebenen Namen (gespeichert in tmp). Hat sie ihn gefunden (if (beginn != -1) liefert true), extrahiert sie den Inhalt des Cookies. Dazu ermittelt die Funktion die Positionen des ersten und letzten Zeichens des Cookie-Inhalts. Die Position des ersten Zeichens ist die Position hinter dem Namen des Cookies, die Position des letzten Zeichens findet man, wenn man mit indexOf() nach dem abschließenden Semikolon sucht. Der String zwischen beiden Positionen wird dann als Teilstring zurückgeliefert, decodiert (Aufruf von unescape()) und in der Variablen ergebnis gespeichert.

## Testen

Zum Austesten ihrer Cookie-Funktionen sollten Sie sich eine kleine Testwebseite mit einem Formular zum Setzen und Abfragen der Cookies erstellen.

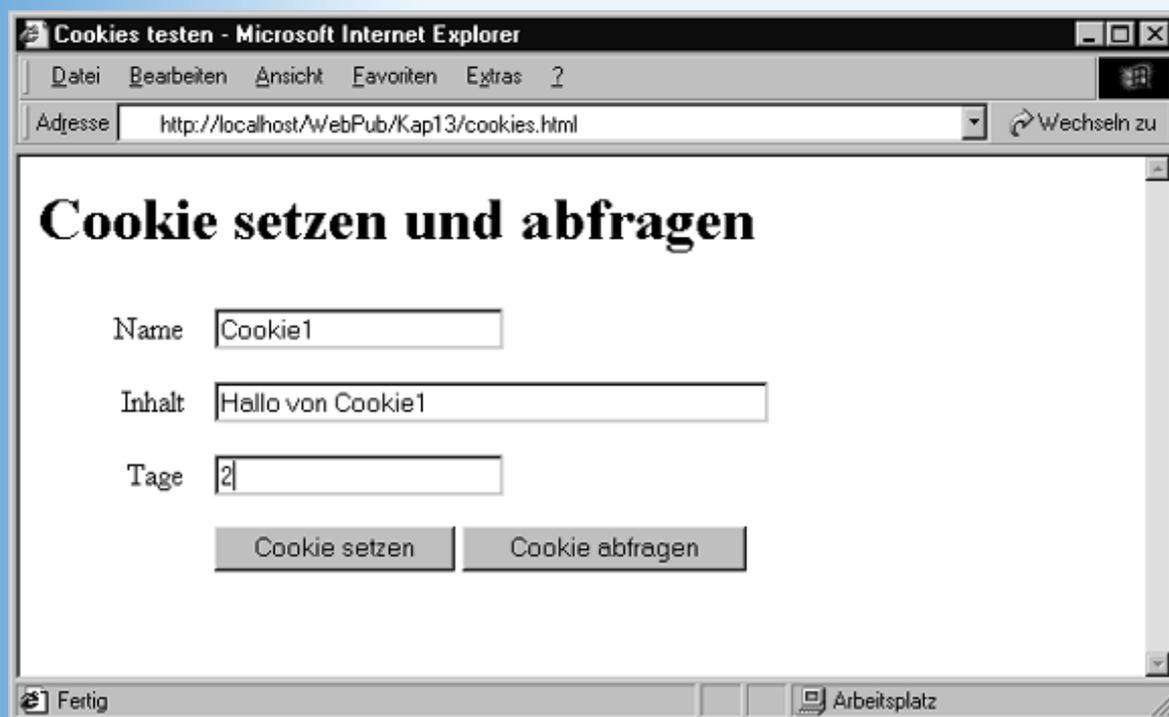


Abbildung 13.1: Cookie-Testseite

Um einen Cookie zu setzen, füllen Sie das Formular zuerst vollständig aus (siehe Abbildung 13.1) und drücken dann die Schaltfläche **Cookie setzen**. Setzen Sie zur Probe ruhig mehrere Cookies!

Um danach den Inhalt eines Cookies abzufragen, geben Sie nur den Namen des Cookies ein und klicken Sie dann auf die Schaltfläche **Cookie abfragen**. Danach sollte im Inhalt-Feld der Inhaltstext des Cookies angezeigt werden.

Hier noch einmal der vollständige Code dieser Seite.

### Listing 13.1: cookies.html - Cookies-Mechanismus testen

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Cookies testen</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">

```

```

function CookieSetzen(name, inhalt, tage)
{
 var cookieDate = new Date();
 cookieDate.setTime(cookieDate.getTime()
 + (1000 * 60 * 60 * 24 * tage));
 document.cookie = name + "=" + escape (inhalt) +
 "; expires=" + cookieDate.toGMTString();
 // Formular zurücksetzen
 document.forms[0].elements[0].value = "";
 document.forms[0].elements[1].value = "";
 document.forms[0].elements[2].value = "";
}
function CookieAbfragen(name)
{
 var ergebnis="";
 var tmp = name + "=";
 if (document.cookie.length > 0)
 {
 beginn = document.cookie.indexOf(tmp)
 if (beginn != -1)
 {
 beginn += tmp.length;
 ende = document.cookie.indexOf(";", beginn);
 if (ende == -1)
 ende = document.cookie.length;
 ergebnis=
 unescape(document.cookie.substring(beginn, ende));
 }
 }
 document.forms[0].elements[1].value = ergebnis;
}
</script>
</head>
<body>
<h1>Cookie setzen und abfragen</h1>
<form>
<table border="0" width="400" cellspacing="5" cellpadding="5">
<tr>
<td width="67" align="right">Name</td>
<td width="291"><input type="text" name="T1"
size="20"></td>
</tr>
<tr>
<td width="67" align="right">Inhalt</td>
<td width="291"><input type="text" name="T2"
size="40"></td>
</tr>
<tr>
<td width="67" align="right">Tage</td>
<td width="291"><input type="text" name="T3"
size="20"></td>
</tr>
<tr>
<td width="67" align="right"></td>
<td width="291">
<input type="button" value="Cookie setzen" name="B1"
onClick=
"CookieSetzen(document.forms[0].elements[0].value,

```

```

 document.forms[0].elements[1].value,
 document.forms[0].elements[2].value) ">
<input type="button" value="Cookie abfragen" name="B2"
onClick=
"CookieAbfragen(document.forms[0].elements[0].value) "></td>
</tr>
</table>
</form>
</body>
</html>

```

## 13.3 Cookies löschen

Wie Sie bereits wissen, verfallen Cookies automatisch nach Ablauf ihres Gültigkeitsdatums. Es gibt allerdings auch einmal Situationen, in denen man einen Cookie direkt löschen möchte.

Wenn Sie einen Cookie gleichen Namens mehrmals erzeugen, werden nicht mehrere Instanzen des gleichen Cookies angelegt. Vielmehr gibt es auf dem Client-Rechner stets nur einen Cookie dieses Namens, der bei jeder Erstellung eines neuen Cookies gleichen Namens überschrieben wird.

Dies nutzt man zum Löschen von Cookies. Erzeugen Sie einfach unter dem Namen des zu löschenden Cookies einen neuen Cookie, dessen Zerfallsdatum Sie in die Vergangenheit zurücklegen. Der Cookie wird dann überschrieben - und sofort vom Client gelöscht, da er ja bereits abgelaufen ist.

```

function CookieLoeschen(name)
{
 var cookieDate = new Date();
 cookieDate.setTime(cookieDate.getTime() - 1);
 document.cookie = name + "=Nur zum Loeschen" +
 "; expires=" + cookieDate.toGMTString();
}

```



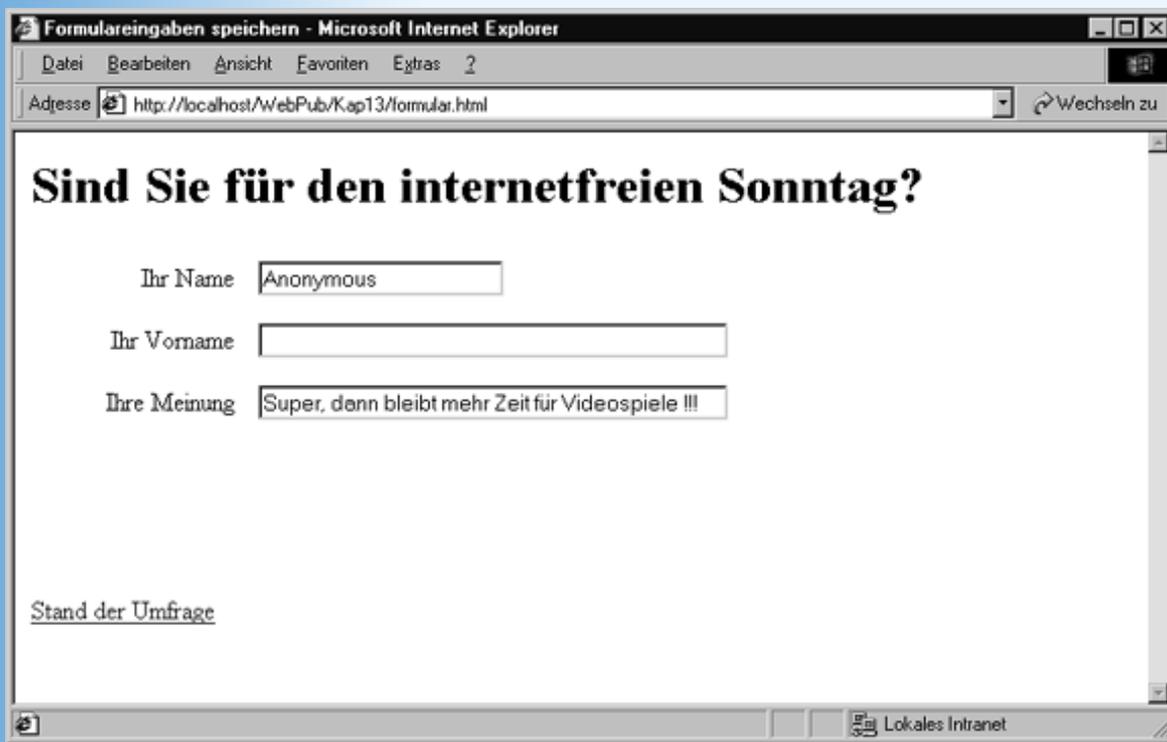
*Versuchen Sie nicht mit einem Cookie zu überschreiben, der kein Zerfallsdatum hat. Diese Cookies, die nur für die aktuelle Sitzung mit dem Browser gültig sind, werden überhaupt nicht abgespeichert, können also auch nicht bereits gesetzte Cookies überschreiben oder gar löschen.*

## 13.4 Einstellungen einer Seite speichern

Ein häufiges Problem bei der Erstellung von dynamischen Webseiten ist, dass es keine Möglichkeit gibt, den Zustand einer Webseite bis zum nächsten Aufruf der Webseite festzuhalten.<sup>2</sup>

Stellen Sie sich vor, Sie präsentieren auf einer Seite Ihrer Website eine Bildergalerie - allerdings nicht als scrollbare Bilderflut, sondern als Diashow, die über Vor- und Zurück- Schalter gesteuert wird. Wenn nun der Besucher Ihrer Website die Seite mit der Diashow mitten in der Bilderfolge verlässt, um nach kurzer Zeit wiederzukehren, wird er enttäuscht sein, wenn er alle Bilder bis zu der Position, an der er angekommen war, noch einmal durchblättern muss.

Gleiches gilt für das Ausfüllen von Formularen. Wenn der Besucher die Seite mit einem halb ausgefüllten Formular kurzzeitig verlässt, ist beim Zurückkehren von seinen bisher gemachten Eintragungen nichts mehr zu sehen (außer er verwendet nur die Vor- und Zurück-Schalter seines Browser).



**Abbildung 13.2: Rekonstruktion von Formulareingaben**

Mit JavaScript und Cookies kann man hier Abhilfe schaffen. Die folgenden beiden Funktionen speichern und rekonstruieren den Zustand eines einfachen Formulars mit drei Texteingabefeldern.

```
function CookieFormularSpeichern(name)
{
 var feld0 = document.forms[0].elements[0].value;
 var feld1 = document.forms[0].elements[1].value;
 var feld2 = document.forms[0].elements[2].value;

 document.cookie = name + "=" + escape(feld0) + "|"
 + escape(feld1) + "|" + escape(feld2)
}

```

Die Funktion `CookieFormularSpeichern()` liest die Werte aus den Formularfeldern aus und speichert sie als |-getrennte Liste in einem Cookie. (Setzt natürlich voraus, dass in den Formulareingaben keine senkrechte Strichzeichen | verwendet werden.)

```
function CookieFormularLesen(name)
{
 name += "=";
 var laenge = name.length;
 var cookie_laenge = document.cookie.length;
 var i = 0;
 while (i < cookie_laenge)
 {
 var j = i + laenge;
 if(document.cookie.substring(i,j) == name)
 {
 return WerteAuslesen(j);
 }
 i = document.cookie.indexOf(" ", i) + 1;
 if (i == 0)
 {
 break;
 }
 }
}

```

```

 }
 return null;
}

```

Die Funktion zum Auslesen ist hier etwas umfangreicher, da man berücksichtigen muss, dass unter Umständen mehrere Cookies für die Webseite gültig sind.

In so einem Fall liefert der Browser alle Cookies durch Leerzeichen getrennt zurück und es ist Aufgabe des JavaScripts, den Cookie herauszufiltern, der die Eingaben des Formulars enthält. Dies leistet die Funktion `CookieFormularLesen()`. Wenn Sie den Cookie für das Formular gefunden hat, übergibt sie diesen an die Hilfsfunktion `WerteAuslesen()`, deren Aufgabe es ist, die einzelnen Feldinhalte auszuwerten und auf die Formularfelder zu verteilen.

```

function WerteAuslesen(beginn)
{
 var naechster = document.cookie.indexOf(";", beginn);
 if (naechster == -1)
 naechster = document.cookie.length;
 var i = 0;
 while (beginn < naechster)
 {
 ende = document.cookie.indexOf("|", beginn);
 if (ende == -1)
 ende = naechster;
 document.forms[0].elements[i].value =
 unescape(document.cookie.substring(beginn, ende));
 beginn = ende + 1;
 i += 1;
 }
 return 1;
}

```

Aufgerufen werden die beiden Funktionen zum Rekonstruieren und Speichern der Formulareingaben als Antwort auf die `onload`- und `onunload`-Ereignisse der Webseite.

### Listing 13.2: Auszug aus `formular.html`

```

...
<body onload="CookieFormularLesen('Formulardaten')"
 onunload="CookieFormularSpeichern('Formulardaten')">
<h1>Sind Sie für den internetfreien Sonntag?</h1>
<form>
 <table border="0" width="450" cellspacing="5" cellpadding="5">
 <tr>
 <td width="150" align="right">Ihr Name</td>
 <td width="300"><input type="text" name="T1"
 size="20"></td>
 </tr>
 <tr>
 <td width="150" align="right">Ihr Vorname</td>
 <td width="300"><input type="text" name="T2"
 size="40"></td>
 </tr>
 <tr>
 <td width="150" align="right">Ihre Meinung</td>
 <td width="300"><input type="text" name="T3"
 size="40"></td>
 </tr>

```

```
</table>
</form>
<div style="padding-top: 5em">
Stand der Umfrage
</div>
</body>
</html>
```



*Die Seite `umfrage.html`, die sich ebenfalls auf der Buch-CD befindet, liefert keine echte Auswertung der Formulareingaben. Sie dient nur zum bequemen Verlassen und Wiederansteuern der Webseite `formular.html`. Wie man Gästebücher anlegt oder Umfragen auswertet, erfahren Sie in Woche 3.*

## 13.5 Cookies, Virtuelle Warenkörbe und CGI

Die im vorangehenden Abschnitt vorgestellte Technik eignet sich natürlich nicht nur dazu, dynamische Webseiten gemäß den letzten Einstellungen wiederherzustellen.

Genauso gut können Sie mit Cookies und JavaScript auch Daten zwischen verschiedenen Webseiten eines Webs austauschen - eine Technik, für die es wiederum eine Vielzahl von Anwendungsmöglichkeiten gibt, beispielsweise die Erstellung virtueller Warenkörbe.

### Pfadangaben für Cookies

Wenn man vermittels Cookies Daten zwischen verschiedenen Seiten eines Webs austauschen will, muss man darauf achten, dass die Cookies auch für alle betroffenen Webseiten gültig sind.

Man erreicht dies, indem man

- entweder alle Webseiten in einem gemeinsamen Verzeichnis unterbringt (manche Webserver zwingen einen ja geradezu dazu, doch empfehlenswert ist es natürlich nicht)
- oder Sie setzen die Pfadangabe des Cookies auf ein übergeordnetes Verzeichnis, in dessen Unterverzeichnissen die Webseiten enthalten sind ("; path=/meinWeb/").

Wenn Sie über keine eigene Domain verfügen, sollten Sie als Pfad zumindest den Pfad zu Ihren Verzeichnissen eingeben und nicht etwa Angaben wie "/" verwenden, die den Cookie für alle Unterverzeichnisse der Domain verfügbar machen.



*Virtuelle Warenkörbe lassen sich zwar vollständig mit JavaScript realisieren, doch üblich ist es eigentlich nicht, jedenfalls nicht für umfangreichere Warenangebote, die datenbankgestützt sind. Da hierbei für den Aufbau der Webseiten ehemals serverseitige Techniken eingesetzt werden (CGI, ASP, etc.) wird üblicherweise auch der Warenkorb serverseitig per CGI realisiert.*

## 13.6 Zusammenfassung

Cookies stellen einen vom Browser kontrollierten Mechanismus zum clientseitigen Abspeichern von Daten dar. In JavaScript kann man das vom Browser zur Verfügung gestellte Objekt `document.cookie` und dessen Eigenschaften

und Methoden zum Setzen und Abfragen von Cookies verwenden.

## 13.7 Fragen und Antworten

**Frage:**

**In der Einleitung zu diesem Kapitel wurde erwähnt, dass man mit Hilfe eines Cookies feststellen kann, ob ein Websurfer eine Website schon einmal betreten hat und dementsprechend auf die Anzeige einer Eingangsseite verzichten kann. Muss oder sollte man einen solchen Cookie implementieren, wenn man vor die zentrale Homepage des Webs eine Eingangsseite schaltet?**

*Antwort:*

*Nein, nur weil Sie eine Eingangsseite für Ihr Web vorsehen, müssen Sie keinen Cookie setzen, der dafür sorgt, dass die Eingangsseite bei neuerlichem Besuch des Webs übersprungen wird. Wenn ein Besucher von Ihrer Webseite so begeistert ist, dass er sie in der Zukunft noch öfter besuchen wird, sollte er die zentrale Homepage in die Liste seiner Favoriten aufnehmen. Dann kann er die Homepage direkt ansteuern und die Eingangsseite wird ehedem nicht angezeigt. Insofern kann eine Eingangsseite - oder gar ein Eingangstunnel - auch ein zusätzlicher Anreiz sein, die Homepage in die Favoritenliste einzutragen. ;-)*

**Frage:**

**Ist es sinnvoll, eine Webseite so aufzusetzen, dass sie nur dann funktioniert, wenn ein Cookie angelegt und ausgewertet wird?**

*Antwort:*

*Cookies sind eine Art kleine, dienstbare Geister, mit deren Hilfe man Webseiten interaktiver und benutzerfreundlicher machen kann. Keinesfalls aber sollte man eine Webseite so konstruieren, dass sie auf die Unterstützung von Cookies angewiesen ist. Denn erstens verwenden die einzelnen Browser unterschiedliche Mechanismen zum Abspeichern von Cookies (das heißt, Cookies, die von einem Browser gesetzt werden, bleiben den anderen Browsern verborgen), und zweitens gibt es viele Websurfer (und Firmen), die das Anlegen von Cookies aus Sicherheitserwägungen heraus abgeschaltet haben.*

## 13.8 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

1. Welches JavaScript-Objekt repräsentiert die Cookies zu einer Webseite?
2. Wie kann man einen neuen Cookie für eine Webseite setzen?
3. Wie kann man den Inhalt eines Cookies abfragen?
4. Wie kann man einen Cookie für mehrere Webseiten setzen?

### Übungen

1. Wenn Sie sich das Beispiel aus Abschnitt 13.4 anschauen, werden Sie feststellen, dass die Seite mit den Umfrageergebnissen nicht so recht zu dem Umfrageformular passt, da das Umfrageergebnis als eine Pro/Kontra-Statistik präsentiert wird, während das Formular lediglich die Eingabe von Kommentaren erlaubt. Nun könnte man zwar versuchen, aus den Kommentaren die Einstellung des Absenders herauszulesen, doch lässt sich dies kaum mit Hilfe eines Programms lösen (irgend jemand müsste also regelmäßig die abgeschickten Kommentare lesen, auswerten und die Seite mit dem Umfrageergebnis aktualisieren). Hätte das Umfrageformular dagegen zwei Optionsfelder »Pro« und »Kontra« könnte man die Formulareingaben serverseitig auswerten und die Webseite mit dem Umfrageergebnis von einem Programm aktualisieren lassen. Wie so etwas geht, werden Sie allerdings erst in der dritten Woche erfahren (siehe Kapitel 17-20).

Hier begnügen wir uns mit der clientseitigen Unterstützung. Erweitern Sie das Formular um zwei passende Optionsfelder und passen Sie den JavaScript-Code für die Abspeicherung und

Rekonstruktion der Formulareingaben an.

Formulareingaben speichern - Microsoft Internet Explorer

Adresse [http://localhost/WebPub/Kap13/Uebungen/Ueb13\\_01/formular.html](http://localhost/WebPub/Kap13/Uebungen/Ueb13_01/formular.html) Wechsell zu

## Sind Sie für den internetfreien Sonntag?

Ihr Name

Ihr Vorname

Pro  Kontra

Ihre Meinung

[Stand der Umfrage](#)

Fertig Lokales Intranet

Abbildung 13.3: Erweitertes Formular

---

❖ Kapitel Inhalt Index **SAMS** Top Kapitel ❖

---

1 Mit Client-Rechner meinen wir den Rechner des Websurfers, der unsere Webseiten besucht und in seinem Browser anschaut.

2 Dies hat damit zu tun, dass Webseiten mittels des HTTP-Protokolls über das Netz übertragen werden. Das HTTP-Protokoll ist ein statusloses Protokoll, das keine Zwischenspeicherung von Daten zulässt.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

## Tag 14

# Erste serverseitige Techniken

Zum Ende dieser Woche wollen wir die Client-Seite verlassen und uns ein wenig auf der Serverseite umtun.

Die Themen heute:

- JavaScript auf der Serverseite
- Dynamische Inhalte mit Server Side Includes

## 14.1 Serverseitiges JavaScript

Ja, Sie haben richtig gelesen. Mit JavaScript, der Standardsprache für clientseitige Programmierung, kann man auch serverseitige Anwendungen schreiben.

Serverseitiges JavaScript verwendet die gleiche Syntax wie clientseitiges JavaScript (logisch, es ist ja nach wie vor die gleiche Programmiersprache), nicht aber die gleichen vordefinierten Objekte. An die Stelle von window, document, location, etc. treten auf der Serverseite Objekte wie server, projekt, client, request oder file, mit deren Hilfe man beispielsweise Formulareingaben bearbeiten, Dateien öffnen und speichern oder auf Datenbanken zugreifen kann - alles wie gesagt auf der Serverseite!

Wir werden in diesem Buch allerdings nicht näher auf die serverseitigen Objekte oder die Möglichkeiten der serverseitigen JavaScript-Programmierung eingehen, da JavaScript auf der Serverseite bei weitem nicht die gleiche Bedeutung zukommt, wie bei der clientseitigen Programmierung. Die Serverseite, das ist die Domäne von Sprachen wie Perl oder C++ und von Technologien wie ASP (Active Server Pages) oder PHP, mit denen wir uns in der dritten Woche befassen werden.

Wer dennoch Javascript zur serverseitigen Programmierung einsetzen möchte, der braucht zuerst einmal einen entsprechenden Webserver, der serverseitiges JavaScript unterstützt. Hier käme vor allem der Enterprise-Server von Netscape in Frage. Microsofts IIS-Server unterstützt serverseitiges JavaScript im Rahmen seiner Active Server Pages, wobei anzumerken ist, dass ASP traditionell zumeist mit VBScript, dem Microsoft-eigenen Konkurrenzprodukt zu JavaScript, programmiert wird (siehe Kapitel 18).

## 14.2 Server Side Includes



*Server Side Includes (SSI) sind besondere, in einem HTML-Dokument eingebettete Kommentare, die als Platzhalter für spezielle Inhalte fungieren, die dynamisch beim Aufruf der betreffenden Webseite eingefügt werden.*

Fordert ein Webbrowser von einem Webserver ein HTML-Dokument an, das Server Side Includes enthält, schickt der Webserver die Datei nicht gleich wie sonst an den Browser. Statt dessen liest er das HTML-Dokument zuerst ein, sucht nach den darin enthaltenen Server Side Includes und wertet diese aus - was bedeutet, dass er die SSIs durch einen dynamisch erzeugten Inhalt ersetzt. Das resultierende HTML-Dokument schickt er dann an den Browser. Voraussetzung dafür, dass alles wie gewünscht funktioniert, ist allerdings

- dass der Webserver Server Side Includes unterstützt,
- dass der Webserver korrekt zur Verarbeitung von Server Side Includes konfiguriert ist,
- dass die Webseiten SSIs enthalten und die korrekte Dateierweiterung aufweisen.



*Es gibt viele verschiedene Implementierungen von Server Side Includes, die sich in ihrer Komplexität stark unterscheiden. Wir werden uns im Folgenden auf die Server Side Includes beschränken, die von den meisten Webservern, insbesondere Apache, PWS und Omni HTTPd, unterstützt werden.*

## **Konfiguration des Webserver**

Um die Auswertung von Server Side Includes zu aktivieren, müssen Sie in zwei Schritten vorgehen.

1. Zuerst müssen Sie dem Server angeben, welche Dateien auf SSIs hin untersucht und entsprechend verarbeitet werden sollen.

Die übliche Erweiterung für Webseiten mit Server Side Includes lautet `.shtml` (oder auch `.shtm` oder `.stm`).

Wenn Sie wollen, können Sie als Dateierweiterung auch `.html` und/oder `.htm` angeben, so dass alle Webseiten auf dem Server nach SSIs durchsucht und ausgewertet werden. Allerdings sollte man bedenken, dass das Durchsuchen der HTML-Dokumente die Performance des Webserver herabsetzt (schließlich muss der Server die Webseiten analysieren und die enthaltenen SSIs auswerten, bevor er sie an die Browser schicken kann). Wenn Sie als Extension `.html` wählen, ist dies zwar recht bequem, bedeutet aber auch, dass der Webserver unter Umständen unnötigerweise viele Webseiten parsen muss, die gar keine SSIs enthalten. Zudem entsteht ein nicht zu unterschätzendes Sicherheitsloch (siehe Frage/Antwort-Teil)



*Wenn Sie Ihre Webseiten auf dem Webserver Ihrer Firma oder eines Internet Providers veröffentlichen, können Sie die Extension natürlich nicht selbst festlegen, sondern müssen bei dem verantwortlichen Webadministrator nachfragen, welche Extension Sie Ihren SSI-Webseiten geben müssen.*

2. Danach müssen Sie noch die Unterstützung von Server Side Includes einschalten, damit der Webserver die Webseiten mit den angegebenen Extensionen auch wirklich analysiert und die gefundenen Server Side Includes auswertet.

Schauen wir uns kurz an, wie unsere drei Beispielwebserver für die Verarbeitung von Server Side Includes konfiguriert werden.

## Der Apache-Server

Das Apache-Server ist standardmäßig meist so konfiguriert, dass er keine Server Side Includes unterstützt. Das hat zum einem damit zu tun, dass die Auswertung der Server Side Includes die Performance des Webservers herabsetzt, zum anderen damit, dass SSIs für einen Webserver auch ein gewisses Sicherheitsrisiko darstellen (dann nämlich, wenn man den SSIs auch das Ausführen von Systembefehlen erlaubt, siehe unten).

Um die Auswertung von Server Side Includes zu aktivieren, gehen Sie wie folgt vor.

1. Um festzulegen, welche Dateien nach Server Side Includes durchsucht werden sollen, müssen Sie die Datei `httpd.conf` (`srm.conf` für ältere Versionen des Webservers) öffnen und um folgenden Eintrag erweitern:

```
AddType text/x-server-parsed-html .shtml
```

Der Server wird jetzt jede Datei mit der Extension `.shtml` nach Server Side Includes durchsuchen. Wenn Sie wollen, können Sie mittels einer zweiten Zeile eine zweite Extension (beispielsweise `.shtm`) angeben.

2. Jetzt müssen Sie noch die Server Side Includes aktivieren.

Im Falle des Apache-Servers können Sie festlegen, welche Gruppen von Server Side Includes ausgeführt werden sollen. Die wichtigsten Gruppen sind Includes (Einfügen von Webseiten, Umgebungsvariablen, statistischen Daten) und ExecCGI (Ausführung von CGI-Programmen und Systembefehlen, deren Ausgaben in die Webseite eingefügt werden). Aus Sicherheitsgründen sollte man sich bei nicht lokalen Webservern auf die Includes-SSIs beschränken.

Öffnen Sie die Datei `httpd.conf` (`access.conf` für ältere Versionen des Webservers). Fügen Sie die Includes-Option in die Zeile Options ein, beispielsweise:

```
Options Includes
```

oder

```
Options IncludesNOEXEC ExecCGI
```

oder

## Der Personal Webserver

Der Personal Webserver ist standardmäßig so eingerichtet, dass er Server Side Includes in Webseiten mit den Extensionen .shtml, .shtm und .stm unterstützt.

Unter Windows NT/2000 kann man über den Internetdienst-Manager (siehe Anhang A) eigene Extensionen festlegen. Unter Windows 95/98 geht dies nur durch direkte Manipulation der Systemregistrierung.

Damit der Webserver die Dateien mit den Server Side Includes wie gewünscht auswertet, müssen Sie gegebenenfalls noch sicherstellen, dass für die Verzeichnisse, in denen die HTML-Dokumente mit den SSIs liegen, Skript- und Ausführungsberechtigung gesetzt sind.

## Der OmniHTTPd-Server

Der OmniHTTPd-Webserver ist standardmäßig so eingerichtet, dass er Server Side Includes in Webseiten mit der Extension .shtml unterstützt.

Um eine andere Dateierweiterung festzulegen, rufen Sie über die Programmgruppe des OmniHTTPd-Webserver das Administrationsprogramm auf. Klicken Sie auf den Schalter **Web Server Global Settings** und wechseln Sie in dem erscheinenden Dialog zur Registerkarte **MIME**. Scrollen Sie in dem Listenfeld zu dem Eintrag:

wwwserver/html-ssi .shtml

Markieren Sie den Eintrag, geben Sie im Feld **Actual** eine andere Dateierweiterung ein und klicken Sie auf einen der Schalter zum Hinzufügen (**Add**) oder Ersetzen (**Replace**).

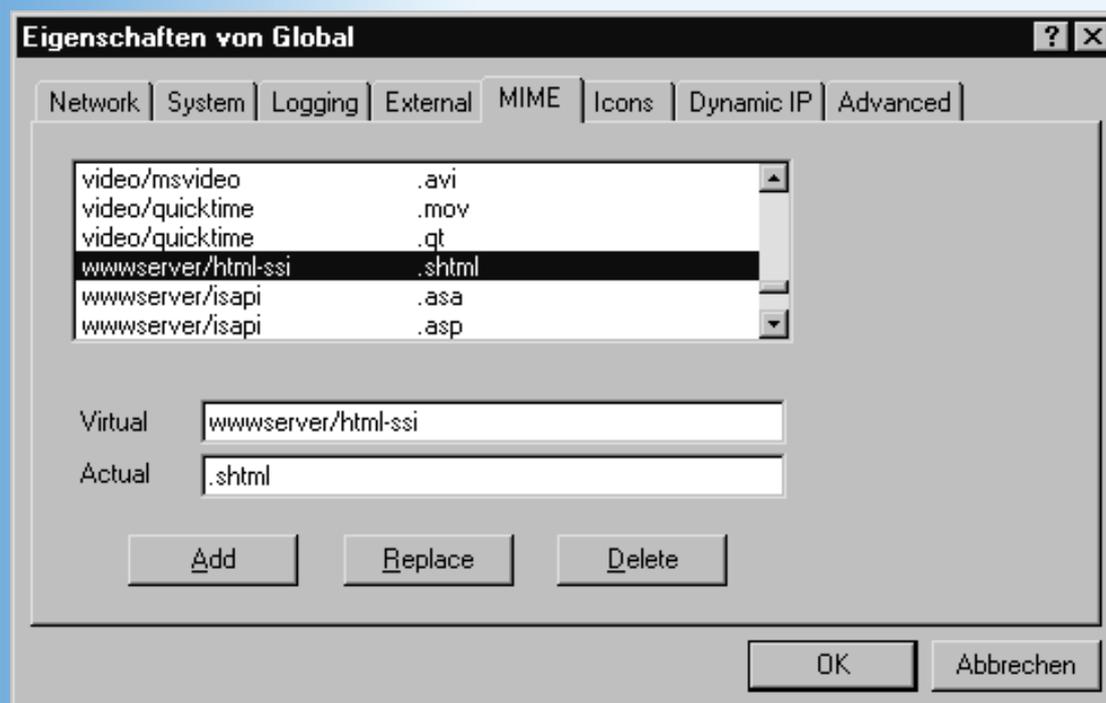


Abbildung 14.1: Dateierweiterung für Webseiten mit Server Side Includes festlegen

Um sicherzustellen, dass der Webserver die Dateien mit den Server Side Includes wie gewünscht auswertet, wechseln Sie zur Registerkarte **Advanced** und vergewissern Sie sich, dass die Option **Process Server Side Includes (SSI)** aktiviert ist.

## Eine Testwebseite

Um die Verarbeitung von Server Side Includes zu testen, speichern Sie in einem Verzeichnis auf Ihrem Webserver folgende beiden Dateien ab.

### Listing 14.1: testssi.dat - Datei, deren Inhalt vom Webserver in eine .shtml-Datei eingefügt wird

```
<p>Wenn Sie diesen Text lesen können, wurde die SSI-Anweisung korrekt ausgeführt!</p>
```

### Listing 14.2: hauptseite.shtml - Webseite mit Server Side Include zum Einfügen von testssi.dat

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Testseite für Server Side Includes</title>
</head>
<body>
<p>Dies ist normales HTML.</p>
<!--#include file="testssi.dat"-->
<p>Dies ist auch normales HTML.</p>
</body>
</html>
```

Wenn Sie die Webseite hauptseite.html als Datei (mit Pfadangabe statt URL) in den Browser laden, sehen Sie nur die beiden Absätze mit dem normalen HTML. Wenn Sie die Webseite dagegen von Ihrem lokalen Webserver laden (siehe Abbildung 14.2) und dieser die Server Side Include-Anweisung korrekt verarbeitet, sehen Sie auch den Text aus der Datei *testssi.dat*, der an der Stelle des Kommentars eingefügt wird.



Abbildung 14.2: Webseite mit vom Server eingefügtem Text

## Anwendungsbeispiele

Was kann man mit Server Side Includes anfangen?

Mit Server Side Includes kann man HTML-Code, Grafiken oder die Werte bestimmter Variablen in Webseiten einfügen - und zwar dynamisch, das heißt, die Inhalte werden jedes Mal neu eingefügt, sowie die Webseite von irgendeinem Browser angefordert wird. Ein typisches Beispiel dafür, wie man dies zum Aufbau dynamischer Webseiten nutzen kann, ist die Einblendung des aktuellen Datums.

## Einbindung von Server Side Includes

### Listing 14.3: datum.shtml - Datumsausgabe mit SSI

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Server Side Includes</title>
</head>
<body>
<h1>Datumsausgabe mit SSI</h1>
<p>Aktuelles Datum: <!--#echo var="DATE_LOCAL" --></p>
</body>
</html>
```



**Abbildung 14.3: Ausgabe von `<!--#echo var="DATE_LOCAL" -->`**

Die Webseite *datum.shtml* ist im Grunde eine ganz normale Webseite, die statt mit der Extension *.html* mit der Extension *.shtml* abgespeichert wird (unter der Voraussetzung, dass der betreffende Webserver so konfiguriert ist, dass er *.shtml*-Dateien auf Server Side Includes hin durchsucht).

In diese Webseite können wir mit Hilfe von HTML-Kommentaren Server Side Includes einbauen, die beim Aufruf der Webseite vom Webserver erkannt und durch einen passenden dynamischen Inhalt ersetzt werden. Alle Server Side Includes sind nach dem gleichen Schema aufgebaut:

```
<!--#ssibefehl parameter="argument"-->
```

*ssibefehl* steht hier für einen von sechs Befehlen: *config*, *include*, *echo*, *fsize*, *flastmod* oder *exec*. Einzelne Webserver stellen zwar noch weitere SSI-Befehle zur Verfügung, doch wollen wir uns hier auf die Befehle konzentrieren, die derzeit von den meisten Servern unterstützt werden und sozusagen das SSI-Grundvokabular bilden.

Auf den SSI-Befehl folgen die Parameter, über die man die Verarbeitung des Befehls genauer steuern kann. Welche Parameter zur Verfügung stehen und welche Werte man diesen als Argument übergeben kann, hängt von dem jeweiligen SSI-Befehl.

Für den SSI-Befehl *#echo* lautet der Parameter beispielsweise *var* und als Wert kann man ihm eine der auf dem Server verfügbaren CGI-Umgebungsvariablen (siehe Kapitel 17) oder eine der folgenden nur für SSI-Seiten verfügbaren Umgebungsvariablen zuweisen.



*Umgebungsvariablen sind »Variablen«, die nicht von einem Programms, sondern vom Betriebssystem des Rechners definiert und erzeugt werden und auf die jedes Programm, das auf dem Rechner ausgeführt wird, abgefragt werden können.*

SSI-Umgebungsvariable	Zweck
DOCUMENT_NAME	Der Name des Dokuments, das der Server zurückliefert
DOCUMENT_URI	Der URI des Dokuments. Hinweis: Dieses ist ein virtueller und kein absoluter URI.
QUERY_STRING_UNESCAPED	Der eigentliche QUERY_STRING, falls einer mit aufgenommen wurde.
DATE_LOCAL	Das örtliche Datum auf dem Webserver
DATE_GMT	Das Datum in westlicher Zeit (GMT).
LAST_MODIFIED	Das Datum, an dem das Dokument zuletzt geändert wurde.

**Tabelle 14.1: SSI-Umgebungsvariablen.**

Mit Hilfe des #echo-Befehls können Sie die Werte dieser Umgebungsvariablen in den HTML-Code einer Webseite einfügen. So fügt zum Beispiel die SSI-Anweisung aus Listing 14.3 das aktuelle Datum in die Webseite ein:

```
<!--#echo var="DATE_LOCAL" -->
```



*Wie Sie bereits aus Kapitel 8.3 wissen, kann man das aktuelle Datum nicht nur per SSI, sondern auch mit Hilfe des JavaScript-Objekts Date in eine Webseite einfügen. Da stellt sich die Frage, welchen Weg man wählen soll. Nun, grundsätzlich ist der JavaScript-Befehl natürlich vorzuziehen, weil er auf dem Client-Rechner ausgeführt wird und daher schneller ist und den Server nicht belastet. Doch nicht immer sind das Date-Objekt und der Server Side Include beliebig austauschbar, denn während Date das aktuelle Datum auf dem Client-Rechner liefert, fügt der Server Side Include das aktuelle Datum auf dem Server-Rechner ein - je nachdem, in welchen Zeitzonen Server- und Client-Rechner liegen, kann das einen Unterschied machen. (Beachten Sie auch, dass manche Server zusätzlich zum Datum die aktuelle Uhrzeit ausgeben.) Überlegen Sie sich also, welches Datum Sie ausgeben wollen.*

## Das Datum der letzten Änderung anzeigen

Nichts ist uninteressanter als heute die Neuigkeiten von gestern zu lesen. Wer im Internet Webseiten mit aktuellem Inhalt präsentiert, sollte es daher nicht versäumen, in den Webseiten zu vermerken, wann diese das letzte Mal überarbeitet und aktualisiert wurden. Dazu braucht man sicherlich keine Server Side Includes, schließlich kann man das Datum der letzten Änderung auch als ganz normalen HTML-Code eingeben.

```
<p>Letzte Änderung: Montag, der 12.05.2000</p>
```

Dies ist akzeptabel, wenn man es mit einer relativ überschaubaren Anzahl von Webseiten zu tun hat,

die in größeren Zeitabständen (beispielsweise einmal im Monat) aktualisiert werden. Steigt jedoch die Anzahl der zu verwaltenden Webseiten, werden die Zeitabstände zwischen den Aktualisierungen kleiner oder tut man sich einfach schwer damit, die Aktualisierung des Datums nicht zu vergessen, ist es besser, die Eintragung des Datums der letzten Aktualisierung zu automatisieren. Mit Hilfe von Server Side Includes ist dies kein Problem:

#### Listing 14.4: lastmodified.shtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Server Side Includes</title>
</head>
<body>
<h1>Das Wetter</h1>
<p>Morgens diesig, in den Niederungen dichter Nebel. Tagsüber Temperaturen
zwischen 4 und 6 Grad Celsisus. Gegen Abend Gewitter. Fazit: Wer heute Sonne
haben will, muss sie im Herzen tragen.
<p>Letzte Änderung: <!--#echo var="LAST_MODIFIED" --></p>
</body>
</html>
```

Wieder verwenden wir den SSI-Befehl `#echo`, diesmal aber mit dem Argument `LAST-MODIFIED`.



*Der Befehl `#echo` mit dem Argument `LAST-MODIFIED` fragt vom Betriebssystem das Datum der letzten Änderung der Webseite ab und fügt es in die Webseite ein. Wenn Sie viel mit dynamischen Inhalten arbeiten, kann es aber sein, dass der aktualisierte Inhalt einer Webseite gar nicht auf Änderungen in der Webseite selbst zurückgeht, sondern auf Änderungen in Dateien, die dynamisch eingebunden werden (beispielsweise eine aktualisierte JavaScript-Datei oder eine mit Hilfe des SSI-Befehls `#include` eingefügte Datei). In so einem Fall können Sie mit Hilfe des SSI-Befehls `#flastmod` angeben, dass nicht das Datum der letzten Änderung der aktuellen Webseite, sondern das Datum der letzten Änderung der eingefügten Datei angezeigt werden soll: `<!--#flastmod file="eineDatei.html" -->`.*

## Modularer Seitenaufbau mit Server Side Includes

Viele Webs sind so aufgebaut, dass die einzelnen Webseiten über gemeinsame Navigations- und Design-Elemente verfügen - beispielsweise ein Banner im oberen Bereich der Webseiten und eine am linken Rand befindliche Navigationsleiste mit Hyperlinks zum Aufrufen der einzelnen Webseiten des Webs. Ein Beispiel für ein solches Web haben wir unter anderem in Kapitel 3, bei der Besprechung der Frames und Inline-Frames, gesehen. Wir greifen hier noch einmal das Beispiel des als Tabelle realisierten Gedichte-Webs aus diesem Kapitel auf. Es bestand aus fünf Webseiten (*tabelle.html*, *panther.html*, *karussell.html*, *herbsttag.html* und *saal.html*), die alle über das gleiche Banner (in unserem Falle eine einfache Überschrift) und die gleiche Navigationsstruktur (eine Inhaltsverzeichnis

aus Hyperlinks) verfügten. Die Hauptseite *tabelle.html* spiegelt diesen Grundaufbau der Seiten wider.

### Listing 14.5: tabelle.html aus Kapitel 3

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Rilke-Gedichte</title>
</head>
<body>
<table border="0" width="700" cellpadding="10" cellspacing="0">
<colgroup span="2">
 <col width="200">
 <col width="500">
</colgroup>
<tr>
 <td colspan="2">
 <h1>Ausgewählte Rilke-Gedichte</h1>
 </td>
</tr>
<tr>
 <td width="200" valign="top">
 <dl>
 <dt>Rilke</dt>
 <dd>Der Panther</dd>
 <dd>Herbsttag</dd>
 <dd>Im Saal</dd>
 <dd>Das Karussell</dd>
 </dl>
 </td>
 <td width="500" valign="top">

 </td>
</tr>
</table>
</body>
</html>
```

Das Problem hierbei ist, dass wir in allen Webseiten den identischen Code für das »Banner« und das Inhaltsverzeichnis stehen haben. Will man ein neues Banner verwenden oder einen Hyperlink auf ein neues Gedicht in das Inhaltsverzeichnis einfügen, muss man alle Webseiten einzeln durchgehen und jedes Mal den Code für Banner und/ oder Inhaltsverzeichnis anpassen.

In Kapitel 3 haben Sie gesehen, wie man sich diese Arbeit durch Verwendung von Frames sparen kann. Hier wollen wir Ihnen noch eine weitere Alternative zeigen: das Einbinden von Dateiinhalten mit Hilfe des `#include`-Befehls.

1. Zuerst gehen wir ähnlich wie bei der Arbeit mit Frames vor: wir lagern den einzufügenden HTML-Code in eine eigene Datei aus. Diese Datei muss allerdings keine HTML-Seite sein und sie muss auch kein vollständiges HTML-Grundgerüst enthalten. Speichern Sie in der Datei lediglich den

HTML-Code, der später vom Webserver an der Stelle des Server Side Includes eingefügt werden soll.

#### Listing 14.6: banner.txt

```
<h1>Ausgewählte Rilke-Gedichte</h1>
```

#### Listing 14.7: inhalt.txt

```
<dl>
 <dt>Rilke</dt>
 <dd>Der Panther</dd>
 <dd>Herbsttag</dd>
 <dd>Im Saal</dd>
 <dd>Das Karussell</dd>
</dl>
```

2. Danach ersetzen wir in allen Webseiten den Code für Banner und Inhaltsverzeichnis durch die entsprechenden Server Side Includes.

#### Listing 14.8: tabelle.html - Einfügen von Dateiinhalten mit #include

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Rilke-Gedichte</title>
</head>
<body>
<table border="0" width="700" cellpadding="10" cellspacing="0">
<colgroup span="2">
 <col width="200">
 <col width="500">
</colgroup>
 <tr>
 <td colspan="2">
 <!--#include file="banner.txt"-->
 </td>
 </tr>
 <tr>
 <td width="200" valign="top">
 <!--#include file="inhalt.txt"-->
 </td>
 <td width="500" valign="top">

 </td>
 </tr>
</table>
</body>
</html>
```



Abbildung 14.4: Webseite mit SSIs für Banner und Inhaltsverzeichnis

## Die wichtigsten Server Side Includes

Die grundlegende Syntax für Server-Side Includes lautet:

```
<!--#ssibefehl parameter="argument"-->
```

*ssibefehl* steht hier für einen von sechs Befehlen:

- #config
- #echo,
- #exec
- #flastmod
- #fsize
- #include

### #config

Mit config können Sie das Verhalten von bestimmten Server-Side Includes konfigurieren. Dies betrifft drei Variablen:

- errmsg
- sizefmt
- timefmt

Die Fehlermeldung *errmsg* wird angezeigt, wenn ein SSI-Fehler aufgetaucht ist.

```
<!--#config errmsg="Server Side Include-Fehler. Wenden Sie sich an Ihren Webadministrator!">
```

In diesem Fall erscheint in Ihrem HTML-Dokument die Meldung »Server Side Include- Fehler. Wenden Sie sich an Ihren Webadministrator!«, wenn ein SSI-Fehler auftritt.

Wenn Sie den SSI-Befehl `fsize` verwenden, um die Größe einer Datei zurückzugeben (siehe unten), können Sie ihn so konfigurieren, dass er den Werte in Bytes angibt:

```
<!--#config sizefmt="bytes" -->
```

Oder Sie konfigurieren ihn mit einer Abkürzung (zum Beispiel Mb für Megabytes oder Kb für Kilobytes):

```
<!--#config sizefmt="abkrz" -->
```

Außerdem können Sie das Format des SSI-Zeitstrings konfigurieren. Wenn Ihre Konfiguration beispielsweise wie folgt aussieht:

```
<!--#config timefmt="%A, %B %d, %Y" -->
```

werden Datumsangaben in folgendem Format ausgegeben:

Sonntag, März 3, 1996

Platzhalter	Bedeutung
%a	Abkürzung für einen Wochentag
%A	Wochentag
%b	Abkürzung für den Monat
%B	Monatsname
%d	Tag
%D	Datum in reinen Zahlen
%H, %l	Stunde (in 24- oder 12-Stundenformat)
%m	Monatsnummer
%M	Minuten
%S	Sekunden
%T	Uhrzeit
%y	zweistellige Jahreszahl
%Y	vierstellige Jahreszahl
%Z	Zeitzone

**Tabelle 14.2: Die wichtigsten Platzhalter für `sizefmt`**<sup>1</sup>

Beachten Sie, dass nicht alle Platzhalter von allen Webservern unterstützt werden.

## #echo

Mit #echo können Sie den Wert einer CGI-Umgebungsvariablen oder einer speziellen SSI-Umgebungsvariablen anzeigen.

SSI-Umgebungsvariable	Zweck
DOCUMENT_NAME	Der Name des Dokuments, das der Server zurückliefert
DOCUMENT_URI	Der URI des Dokuments. Hinweis: Dieses ist ein virtueller und kein absoluter URI.
QUERY_STRING_UNESCAPED	Der eigentliche QUERY_STRING, falls einer mit aufgenommen wurde.
DATE_LOCAL	Das örtliche Datum auf dem Webserver
DATE_GMT	Das Datum in westlicher Zeit (GMT).
LAST_MODIFIED	Das Datum, an dem das Dokument zuletzt geändert wurde.

**Tabelle 14.3: SSI-Umgebungsvariablen.**

## #exec

Mit #exec können Sie die Ausgabe eines CGI-Programms (siehe Kapitel 17) oder eines Systembefehls in eine Webseite einfügen. Zu #exec gibt es zwei Parameter: cgi und cmd. Wenn Sie die Ausgabe eines CGI-Programms mit aufnehmen wollen, verwenden Sie cgi.

```
<!--#exec cgi="/cgi-bin/gaestebuch.cgi" -->
```



*Beim Aufruf eines CGI-Programms über einen Server Side Include kann man keine Aufrufargumente an das CGI-Programm übergeben.*

Wenn Sie einen Systembefehl ausführen wollen, verwenden Sie cmd. Stellen Sie sicher, dass Sie den vollständigen Pfadnamen des Befehls angeben. Um zum Beispiel die Ausgabe des Programms /bin/date mit aufzunehmen, schreiben Sie:

```
<!--#exec cmd="/bin/date" -->
```



Aus Rücksicht auf die Effizienz und Sicherheit des Webserver sollte man keine `#exec-Includes` verwenden.

## #flastmod

Sie können `#flastmod` verwenden, um das letzte Änderungsdatum einer Datei - die entweder als Argument zu einer der Parameter `file` oder `virtual` spezifiziert wurde - anzuzeigen.

```
<!--#flastmod virtual="/index.html" -->
```

Sie können mit `#config` das Format des Datums konfigurieren (siehe oben).

## #fsize

Verwenden Sie `#fsize`, um die Größe einer angegebenen Datei anzuzeigen.

```
<!--#fsize file="hello.html" -->
```

Sie können das Ergebnis des `Include`-Befehls mit Hilfe von `#config` so konfigurieren, dass er den Wert entweder in Bytes oder in abgekürzter Form anzeigt (siehe oben).

## #include

Mit `#include` können Sie entweder eine andere Datei oder, im Falle des Apache-Servers, die Ausgabe eines CGI-Programms mit aufnehmen. `#include` übernimmt einen von zwei Parametern: `file` oder `virtual`. Dabei akzeptiert `file` den Namen einer Datei aus dem aktuellen Verzeichnis oder eine absolute Pfadangabe (was man jedoch vermeiden sollte); `virtual` akzeptiert einen virtuellen Pfad und Dateinamen relativ zu dem Dokumentenwurzelverzeichnis.

Angenommen Sie haben zum Beispiel drei HTML-Dateien: *hallo.shtml*, *gruss.html* und *adresse.html*, und Sie möchten, dass *gruss.html* und *adresse.html* in *hallo.html* aufgenommen werden. Die Dateien befinden sich in dem folgenden virtuellen Verzeichnisbaum (relativ zum Dokumentenwurzelverzeichnis):

```
/adresse.html
/willkommen/hallo.shtml
/willkommen/gruss.html
```

Die Datei *hallo.html* könnte wie folgt aussehen:

```
<!--#include file="gruss.html" -->
<!--#include virtual="/adresse.html" -->
```

Um auf *adresse.html* zuzugreifen, sollten Sie statt `file` die Option `virtual` verwenden, da Sie keine andere Möglichkeit haben, den Verzeichnispfad von *adresse.html* relativ zum aktuellen Verzeichnis *willkommen* auszudrücken. Sie könnten auch Folgendes eingeben:

```
<!--#include virtual="/willkommen/gruss.html" -->
<!--#include virtual="/adresse.html" -->
```

oder

```
<!--#include virtual="gruss.html" -->
<!--#include virtual="/adresse.html" -->
```

## 14.3 Zusammenfassung

Zum Ausklang der zweiten Woche haben wir noch einen kurzen Blick auf erste serverseitige Techniken geworfen. Sie haben erfahren, dass man mit JavaScript auch Programmcode zur Ausführung auf der Serverseite erstellen kann (vorausgesetzt der Webserver unterstützt dies), und Sie haben ein recht einfaches und bequem zu nutzendes Konzept zum serverseitigen Einfügen dynamischer Webinhalte kennen gelernt: die Server Side Includes:

- #config
- #echo,
- #exec
- #flastmod
- #fsize
- #include

## 14.4 Fragen und Antworten

**Frage:**

**Es wurde erwähnt, dass der SSI-Befehl #exec, mit dem man CGI-Programme oder sogar Systembefehle ausführen kann, ein Sicherheitsrisiko für den Webserver darstellt. Wieso ist das so?**

*Antwort:*

*Nehmen wir an, Sie hätten ein CGI-Programm geschrieben, das ein Gästebuch implementiert (siehe Kapitel 17). Das heißt, in Ihrem Web gibt es eine Webseite mit einem Formular, in das der Besucher seinen Namen und seinen Kommentar für das Gästebuch eingeben kann. Schickt er das Formular ab, werden seine Eingaben an das CGI-Programm übergeben. Dieses öffnet darauf die Webseite mit den Gästebucheinträgen, beispielsweise gaestebuch.html, und hängt den neuen Eintrag als formatierten HTML-Code an das Ende der Datei an.*

Nehmen wir weiter an, Sie betreiben einen Linux-Webserver, der alle SSI-Befehle, inklusive #exec, unterstützt und der der Bequemlichkeit wegen so konfiguriert ist, dass er alle HTML-Dateien nach SSIs durchsucht - eine tödliche Kombination. Wenn es jetzt ein Webbesucher darauf anlegt, Ihnen zu schaden, braucht er nur als Kommentar für das Gästebuch einen Server Side Include mit einem Systembefehl zum Löschen von Dateien einzugeben (unter Linux beispielsweise /bin/rm -rf /):

```
<!--#exec cmd="/bin/rm -rf /" -->
```

Wenn danach irgend jemand das Gästebuch anfordert, um die Einträge zu lesen, parst der Webserver die HTML-Datei des Gästebuchs, stößt auf den SSI-Befehl des

Hackers und führt ihn aus.

Deswegen sollte man Webserver, die öffentlich zugänglich sind, nach Möglichkeit so konfigurieren, dass die Ausführung von CGI-Programmen und Systembefehlen per SSI nicht erlaubt wird und dass zumindest nur Dateien mit besonderen Extensionen (beispielsweise .shtml) geparkt und ausgewertet werden.

## 14.5 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

1. Nennen Sie einige bedeutende Programmiersprachen und Technologien zur serverseitigen Unterstützung von Webseiten?
2. Wie fügt man mit Hilfe von SSI das Datum der letzten Änderung einer bestimmten Datei in eine Webseite auf?
3. Warum sollte man den Webserver nicht so konfigurieren, dass er alle Webseiten mit der Extension .html nach Server Side Includes durchsucht?

### Übungen

1. Überarbeiten Sie den Code der Webseite *lastmodified.shtml* aus Listing 14.4 so, dass Sie zur Ausgabe des Datums den SSI-Befehl #flastmod verwenden und das Datum in folgendem Format angezeigt wird:

```
Wednesday, der 10.01.01 (10:28:42)
```

---

❖ Kapitel Inhalt Index **SAMS** ❖ Top Kapitel ❖

---

## Tag 15

# Java-Applets

Java ist eine relativ junge Programmiersprache, die anfänglich vor allem dadurch Furore machte, dass man mit ihrer Hilfe Programme (die sogenannten Applets<sup>1</sup>) für Webseiten erstellen konnte, die clientseitig ausgeführt werden. Mitte der Neunziger war dies eine Sensation. Nicht nur weil die Programme clientseitig ausgeführt wurden (diesen Vorteil konnte auch das ungefähr zeitgleich erscheinende JavaScript verbuchen), sondern weil hinter den Applets eine mächtige Programmiersprache stand, die den Webdesignern Möglichkeiten eröffnete, an die bei Verwendung von JavaScript nicht im Traum zu denken war.

Heute hat sich dieses Bild etwas gewandelt. Zum einen hat sich Java immer mehr als professionelle Programmiersprache etabliert, mit der man nicht nur kleine Applets für Webseiten, sondern auch echte Programme schreiben kann. Zum anderen können viele Webseiten-Elemente, die früher nur mit Java-Applets befriedigend zu realisieren waren, heute mit Hilfe alternativen, einfacher zu erlernenden Technologien (JavaScript, PHP, Flash, etc.) implementiert werden. Trotzdem ist Java nach wie vor eine Schlüsseltechnologie für die Erstellung dynamischer Webseiten, so dass wir Ihnen eine fundierte Einführung in die Erstellung und Nutzung von Applets nicht vorenthalten wollen.

Die Themen heute

- Wie werden Applets ausgeführt?
- Wie werden Applets in Webseiten eingebaut?
- Wie sind Applets aufgebaut?
- Wie implementiert man eigene Applets?
- Wie kann man Parameter an Applets übergeben?
- Wie erzeugt man Grafikausgaben in Applets?
- Wie reagiert man auf Mausklicks in Applets?

## 15.1 Das Applet-Konzept

Bevor wir zu den Applets kommen, möchte ich noch ein paar allgemeine Worte zu Java verlieren.

### Java und der Bytecode

In der Einführung zu JavaScript haben wir bereits angesprochen, dass ausführbare Programme aus einer Folge von in Bitfolgen codierten Maschinencode bestehen. Der vom Programmierer

verfasste Quellcode eines Programms muss daher mit Hilfe eines Interpreters oder Compilers in Maschinencode umgewandelt werden. Das Problem dabei ist, dass die verschiedenen Plattformen (Kombinationen aus Prozessorarchitektur und Betriebssystem: Microsoft Windows/Intel, UNIX/Sparc, MacOS/PowerPC, etc.) unterschiedlichen Maschinencode benötigen.

- Im Falle kompilierter Programmiersprachen (C++) ruft der Programmierer selbst den Compiler auf und erhält als Ergebnis eine EXE-Datei, die er an seine Kunden vertreiben kann. Eine solche EXE-Datei enthält aber plattformspezifischen Maschinencode und kann daher nur auf der Plattform ausgeführt werden, für die sie erstellt wurde. Möchte ein Programmierer sein Programm sowohl an Windows- wie an Linux-Anwender vertreiben, braucht er für beide Plattformen eigene Compiler und muss für beide Plattformen EXE-Dateien erzeugen. (Aus diesem Grunde finden Sie auf Webseiten zum Herunterladen von Software häufig Links für die verschiedenen Plattformen).
- Im Falle interpretierter Programmiersprachen (Perl) gibt der Programmierer seinen Quelltext weiter und baut darauf, dass der Kunde auf seinem System einen passenden Interpreter installiert hat. Will der Kunde das Programm ausführen, ruft er den Interpreter auf und übergibt diesem den Quelltext. Der Interpreter übersetzt dann den Quelltext in plattformspezifischen Maschinencode und führt ihn direkt aus.

Was bedeutet dies für Java-Programme? Java wurde so konzipiert, dass es plattformunabhängig ist. Für Applets ist dies eine absolute Grundvoraussetzung. Wenn Sie ein Applet in eine Webseite einbauen, müssen Sie ja davon ausgehen, dass Ihre Webseiten von Websurfern angefordert werden, die ganz unterschiedliche Rechnerarchitekturen verwenden (manche Websurfer haben Linux-PCs, andere Windows- oder Mac-Rechner).

Daraus folgt, dass Java-Programme interpretiert werden müssen.



*Java-Interpreter kann man sich von der Website <http://java.sun.com> herunterladen.  
Die großen Browser verfügen bereits über eingebaute Java-Interpreter.*

Die Interpretation der Programme bringt aber auch Nachteile mit sich:

- die Ausführung ist langsam, da der Code nicht nur ausgeführt, sondern zuvor erst noch übersetzt werden muss
- der Quellcode wird ausgeliefert und kann von jedermann eingesehen und kopiert werden

Aus diesem Grunde werden Java-Programme auch kompiliert.

Nanu? Java-Programme werden kompiliert und interpretiert? Wie soll das denn gehen? Nun ganz einfach: Der Java-Programmierer lässt das fertige Programm vom Java-Compiler in einen allgemeinen, plattformunabhängigen Maschinencode - den sogenannten Bytecode -

übersetzen. Diesen Code liefert er aus. Der Anwender übergibt diesen Bytecode dem Java-Interpreter, der daraus plattformspezifischen Code erzeugt und diesen ausführt. Der Zwischenschritt mit dem Bytecode hat den Vorteil, dass die Interpretation schneller abläuft und die Anwender statt des Quelltextes nur kryptischen Binärcode zu Gesicht bekommen.

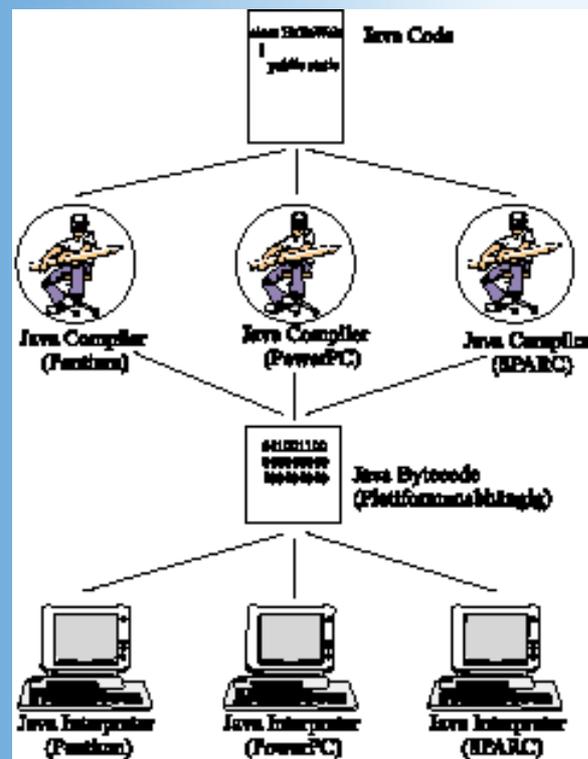


Abbildung 15.1: Erstellung und Ausführung von Java-Programmen

## Erstellung und Ausführung von Applets

Begleiten wir doch einmal ein Applet von seiner Erstellung im Editor des Programmierers bis zur Ausführung im Browser eines Websurfers.

1. Alles beginnt damit, dass irgendein Programmierer oder Webdesigner in seinem Texteditor den Quelltext des Applets aufsetzt und diesen als Textdatei mit der Extension .java abspeichert.



*Der Applet-Quelltext muss in einer eigenen Datei abgespeichert werden, er kann nicht wie JavaScript-Code direkt in den HTML-Code der Webseite eingefügt werden.*

2. Dann ruft der Programmierer den Java-Compiler auf und übergibt ihm die Quelltextdatei. Ist der Quelltext ohne Fehler, erzeugt der Compiler eine Bytecode-Datei mit der Extension .class.
3. Als Nächstes bearbeitet der Programmierer/Webdesigner die Webseite, in die das Applet

eingebettet werden soll. Dazu verwendet er das HTML-Tag <applet> oder <object>.



*In den HTML-Tags gibt man Namen und Ort des Applets an. Im einfachsten Fall kopiert man die CLASS-Datei des Applets direkt in das Verzeichnis, in dem auch die Webseite steht, in der das Applet aufgerufen wird.*

Jetzt machen wir einen Sprung und wechseln zu einem Websurfer, der gerade die Webseite mit dem Applet angesteuert hat.

4. Der Browser lädt die Webseite. Der Webserver schickt ihm neben dem HTML-Code der Webseite und den eingebetteten Bildern auch die class-Datei des Applets.
5. Der Websurfer scrollt den Inhalt der Webseite, bis er zu der Stelle kommt, an der das Applet eingebettet ist.
6. Jetzt ruft der Browser den Java-Interpreter auf (vorausgesetzt er verfügt über einen integrierten Java-Interpreter und die Ausführung von Applets ist nicht deaktiviert) und startet das Applet.

### **Java und die Sicherheit des Client-Rechners**

Wie schon bei JavaScript haben wir auch hier den Fall, dass fremder Code auf dem Rechner der Websurfer ausgeführt wird. Und wieder ist es vor allem der Interpreter, der sicherstellt, dass der Code, in diesem Fall das Applet, auf dem Client-Rechner keinen Schaden anrichten kann. Java-Programmierer sprechen in diesem Zusammenhang von der sandbox, dem »Sandkasten«, in dem sich das Applet tummeln kann, ohne Bereiche außerhalb dieses Sandkastens (andere laufende Anwendungen, Daten auf der Festplatte, etc.) gefährden zu können.

Hierin unterscheiden sich die Applets übrigens auch von den ActiveX-Elementen, die unter MS Windows-Programmierern recht beliebt sind. Diese werden bei Ausführung auf dem Client-Rechner quasi zu einer Erweiterung des Betriebssystems und unterliegen kaum irgendwelchen Beschränkungen.

## **15.2 Eigene Applets erstellen**

Es ist Zeit, mal wieder ein bisschen zu programmieren. Doch bevor wir loslegen können, müssen Sie noch den Java-SDK, die Java-Entwicklungsumgebung von Sun herunterladen und installieren - nicht wegen dem Interpreter (der ist vermutlich bereits in Ihren Browser integriert), sondern wegen des Compilers und den Java-Bibliotheken.

### **Die Java-Entwicklungsumgebung**

Die aktuelle Version finden Sie jeweils auf der Website <http://java.sun.com>, derzeit ist es Java 2, Version 1.3. Laden Sie den SDK herunter und installieren Sie ihn wie auf der Website

beschrieben.

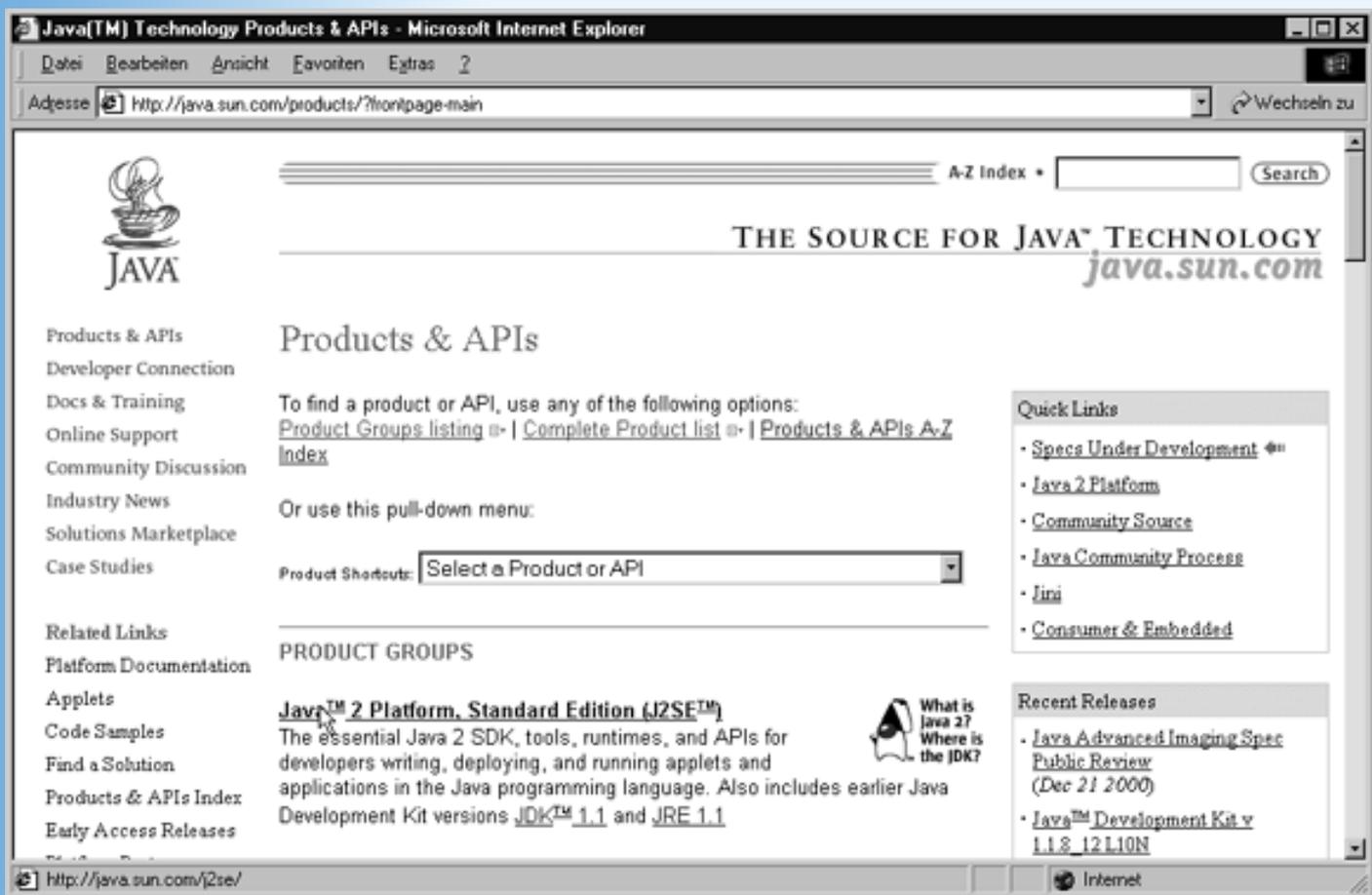


Abbildung 15.2: Die Java-Website von Sun

Zur Entwicklungsumgebung gehören neben den Java-Bibliotheken und einigen Beispielen vor allem die Kommandozeilenwerkzeuge zum Erstellen und Testen der Java-Programme. Diese Programme verfügen über keine grafische Benutzeroberfläche und können nur von der Konsole (Terminal-Fenster) ausgeführt werden.

## Die Windows-Konsole

Im Gegensatz zu Linux-Anwendern wissen viele Windows-Anwender schon gar nicht mehr, was eine Konsole ist.

Unter Windows heißt die Konsole meist MSDOS-Eingabeaufforderung oder ähnlich und kann über das **Start/Programme**-Menü aufgerufen werden. Die Konsole ist rein textbasiert, das heißt sämtliche auszuführenden Befehle müssen Sie über die Tastatur eintippen.

Nach Aufruf der Konsole befinden Sie sich zuerst einmal im Windows-Verzeichnis, beispielsweise C:\Windows. Wenn wir später unsere Java-Programme kompilieren, müssen Sie von dort in das Verzeichnis wechseln, in dem die Quelldatei des Java-Programms steht. Dazu dient der Befehl `cd`.

- Mit `cd ..` wechseln Sie in das übergeordnete Verzeichnis.

- Mit `cd einVerzeichnis` wechseln Sie in ein untergeordnetes Verzeichnis namens »einVerzeichnis«.

Wenn Sie in dem Verzeichnis mit dem Java-Programm angekommen sind, können Sie sich die in dem Verzeichnis enthaltenen Dateien und Unterverzeichnisse durch Eingabe des Befehls `dir` (beziehungsweise `dir /p` für seitenweise Anzeige) anschauen.

Beachten Sie, dass die Konsole lange Dateinamen (mehr als 8 Zeichen ohne Extension) nicht vollständig unterstützt. Wundern Sie sich also nicht, wenn Sie verkrüppelte Namen mit einer Tilde ~ sehen.

1. Rufen Sie jetzt einmal Ihre Konsole auf.
2. Geben Sie am Eingabeprompt `javac` ein und schicken Sie den Befehl durch Drücken der (Return)-Taste ab.

`javac` ist der Java-Compiler. Wenn Sie eine Fehlermeldung erhalten haben, dass der Befehl oder das Programm nicht verfügbar ist, liegt dies daran, dass Ihr System nicht so eingerichtet ist, dass Sie die Java-Werkzeuge von jedem beliebigen Verzeichnis aus aufrufen können. Durchsuchen Sie jetzt Ihre Festplatte nach dem Verzeichnis, in dem `javac.exe` installiert ist. Wenn Sie die Java-Entwicklungsumgebung unter `C:\Java` installiert haben, könnte das Verzeichnis beispielsweise `C:\Java\jdk2\bin` lauten. Wenn Sie jetzt am Prompt zuerst den Pfad zu dem Compiler und dann den Compilernamen eingeben (`C:\Java\jdk2\bin\javac`), müssten Sie als Antwort eine Beschreibung der Aufrufsyntax für den Compiler sehen.

Immer den Pfad mit angeben zu müssen, ist recht lästig. Bequemer ist es, den Pfad zum Compiler und den anderen Java-Werkzeugen in Ihren Systempfad einzutragen.

- Windows 95/98: Laden Sie mit einem Editor die Datei `autoexec.bat` und suchen Sie dort nach einem PATH-Eintrag. Fügen Sie den Pfad zum BIN-Verzeichnis der Java-SDK-Installation hinzu (die einzelnen Pfadangaben im PATH werden durch Semikolons voneinander getrennt):

alter Eintrag: `PATH=.;c:\;c:\dos;c:\windows`

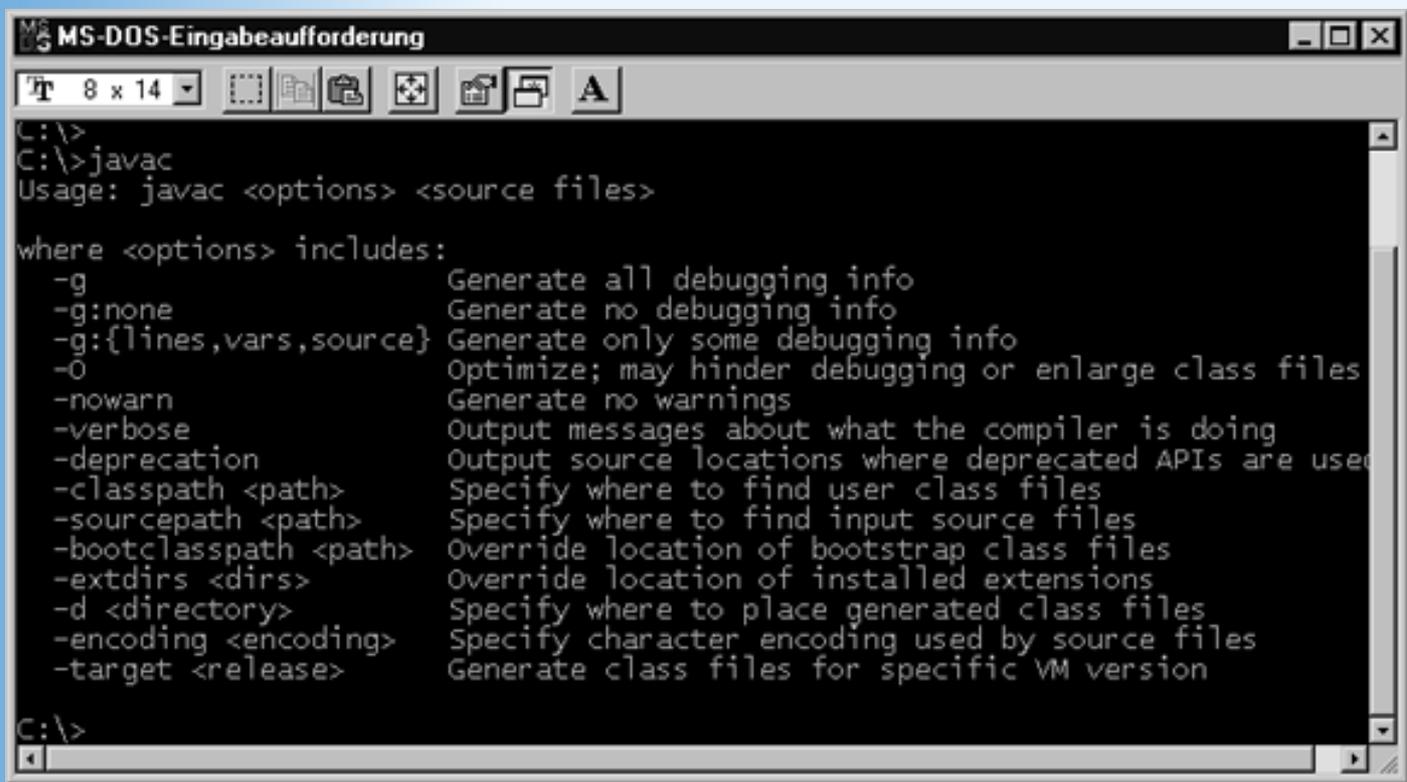
neuer Eintrag: `PATH=.;c:\;c:\dos;c:\windows;c:\ Java\jdk2\bin`

Wenn Sie gar keine PATH-Angabe finden, dann fügen Sie eine neue PATH-Anweisung hinzu.

- Unix: Suchen Sie die Pfadangabe `path` in der zuständigen ini-Datei (je nach Konfiguration `.login`, `.profile`, `.tcshrc` o.ä.) und fügen Sie das Java-Bin-Verzeichnis `/home/myname/jdk2/bin` in der nächsten Zeile nach der bisherigen Pfadangabe hinzu.,

```
set path = (/home/myname/jdk2/bin $path)
```

Nachdem Sie den Computer neu gestartet haben (unter Unix/Linux reicht es, aus- und neu einzuloggen), sollten die Java-Werkzeuge von jedem Verzeichnis aus verfügbar sein (siehe Abbildung 15.3).



```
C:\>
C:\>javac
Usage: javac <options> <source files>

where <options> includes:
-g Generate all debugging info
-g:none Generate no debugging info
-g:{lines,vars,source} Generate only some debugging info
-O Optimize; may hinder debugging or enlarge class files
-nowarn Generate no warnings
-verbose Output messages about what the compiler is doing
-deprecation Output source locations where deprecated APIs are used
-classpath <path> Specify where to find user class files
-sourcepath <path> Specify where to find input source files
-bootclasspath <path> Override location of bootstrap class files
-extdirs <dirs> Override location of installed extensions
-d <directory> Specify where to place generated class files
-encoding <encoding> Specify character encoding used by source files
-target <release> Generate class files for specific VM version

C:\>
```

Abbildung 15.3: javac wurde gefunden

## Ein einfaches Applet

Anhand eines möglichst einfachen Beispiels wollen wir uns nun mit dem Prozess der Applet-Erstellung vertraut machen. Der Code für ein minimales Applets sieht wie folgt aus.

### Listing 15.1: HalloWWW.java

```
// Dies ist das erste Applet
import java.awt.Graphics;
import java.applet.Applet;
public class HalloWWW extends Applet
{
 public void paint(Graphics gc)
 {
 gc.drawString("Hallo World Wide Web!",100,50);
 }
}
```

Versuchen wir zu verstehen, wie dieser Code aufgebaut ist. Lassen Sie sich aber nicht verdrießen, wenn nicht alles klar wird. Java ist eine äußerst leistungsfähige und damit auch eine recht komplexe Programmiersprache. Zudem ist Java rein objektorientiert, was bedeutet, dass sich der Programmierer von Anfang an mit einer Reihe von nicht leicht zu verstehenden, objektorientierten Konzepten auseinandersetzen muss (die Objektorientierung reicht in Java überdies viel weiter, als wir es von JavaScript her kennen). Es wäre also ziemlich vermessen, Java an einem einzigen Tag erlernen zu wollen. Wir beschränken uns daher auf die Vorstellung der wichtigsten Konzepte und einiger ausgesuchter Techniken und Syntaxelemente. Dies sollte ausreichen, eigene Ideen für Applets zu realisieren. Wer sich gründlicher und systematischer in

Java einarbeiten möchte, der sei an die entsprechende Fachliteratur verwiesen (siehe Anhang E).

## Kommentare

Am Anfang unseres Java-Applets steht ein Kommentar. Alle Zeichen einer Zeile, die nach dem // erscheinen, werden von dem Java-Compiler als Kommentar angesehen und ignoriert, da sie nur für den Programmierer als Gedächtnisstütze dienen sollen. Bei kleinen Programmen sind Kommentare noch nicht so wichtig, aber bei etwas größeren Programmierprojekten sind sie unverzichtbar, damit der Programmierer oder andere, die den Code lesen wollen, nachvollziehen können, was das Programm an der jeweiligen Stelle macht.

Mehrzeilige Kommentare kann man mit Hilfe der »Klammern« /\* und \*/ einfügen:

```
/* Dies ist
alles
Kommentar */
```

## Die Hauptklasse des Applets

Die nächsten Zeilen des Beispiels sehen schon ziemlich wild aus und wir werden uns langsam von außen nach innen vorarbeiten.

```
import java.awt.Graphics;
import java.applet.Applet;
public class HalloWWW extends Applet
{

}
```

Hier wird eine Klasse<sup>2</sup> mit dem Namen HalloWWW definiert. Da Java rein objektorientiert ist, bestehen Java-Programme praktisch nur aus Klassendefinitionen. Das muss man erst einmal wirken lassen. In Java kann man Anweisungen nicht wie in JavaScript frei in den Quelltext schreiben. In Java gibt es auch keine Funktionen, sondern nur Methoden (Funktionen, die Teil einer Klasse sind). Wenn wir also mit Java programmieren, definieren wir Klassen mit Datenelementen und Methoden und schreiben alle auszuführenden Anweisungen in die Methoden der Klasse.

Die Definition einer Klasse beginnt mit dem Schlüsselwort class, gefolgt von dem Namen der neuen Klasse (in diesem Beispiel HalloWWW).

```
class HalloWWW
{
}
```

In den Java-Bibliotheken gibt es viele fertige Klassen, die von Java bereitgestellt werden. Eine solche Klasse ist die Klasse Applet, die eine Reihe von Methoden enthält, die für die Applet-

Programmierung wichtig sind: `paint()`, `init()`, `start()`, `stop()` und `destroy()`. Damit wir diese Methoden auch für unsere Applet-Klasse `HalloWWW` nutzen können, geben wir Applet als Basisklasse unserer Applet-Klasse an.



*Tatsächlich wird unser Programm überhaupt erst dadurch zu einem Applet, dass wir die Hauptklasse des Programms von der Basisklasse `Applet` ableiten.*

```
class HalloWWW extends Applet
{
}
```

Das Schlüsselwort `extends` teilt dem Compiler mit, dass unsere Klasse `HalloWWW` alle Elemente der angegebenen Basisklasse (`Applet`) erben soll.

Jetzt gibt es nur noch ein Problem. Die Java-Bibliotheken sind sehr umfangreich und enthalten einige hundert Klassen. Würde der Compiler den Programmen automatisch alle Klassen aus diesen Bibliotheken zur Verfügung stellen, müsste er für jedes Programm standardmäßig Hunderte von Namen verwalten. Aus diesem Grunde stellt der Compiler nur einige wenige Klassen automatisch zur Verfügung, den Rest muss man explizit importieren. Dies geschieht mit Hilfe der `import`-Anweisungen am Anfang des Programms:

```
import java.awt.Graphics;
import java.applet.Applet;
class HalloWWW extends Applet
{
}
```



*Die Klassen der Java-Bibliothek sind in Paketen (englisch `packages`) organisiert. Benötigt man mehrere Klassen eines solchen Pakets braucht man die Klassen nicht einzeln zu importieren. Mit Hilfe des `*`-Zeichens kann man alle Klassen eines Pakets auf einmal importieren. So importiert beispielsweise die Anweisung `import java.awt.*` alle Klassen aus dem `awt`-Paket.*

Zu guter Letzt müssen wir der Klassendeklaration noch das Schlüsselwort `public` voranstellen:

```
import java.awt.Graphics;
import java.applet.Applet;
```

```
public class HalloWWW extends Applet
{
}
```

Das Schlüsselwort `public` ist einer von drei Zugriffsmodifizierern (`public`, `protected`, `private`), die regeln, inwieweit andere Teile des Codes auf diese Klasse zugreifen können. Die Hauptklasse eines Applets muss unbedingt mit dem Modifizierer `public` deklariert werden - sonst hätte der Browser keine Berechtigung das Applet, das er mit der Webseite geladen hat, auszuführen.

## Die Applet-Methoden

Auch wenn es vielleicht gar nicht so aussieht, aber unsere Klasse `HalloWWW` enthält jetzt - dank der Ableitung von der Klasse `Applet` - tatsächlich bereits fünf Methoden namens `paint()`, `init()`, `start()`, `stop()` und `destroy()`. Und das ist auch gut so, denn wenn der Browser das Applet ausführt, ruft er genau diese Methoden auf:

- `init()`, wird aufgerufen, nachdem der Browser das Applet geladen hat
- `start()`, wird aufgerufen, wenn der Browser das Applet startet
- `paint()`, wird aufgerufen, wenn der Browser möchte, dass sich das Applet selbst zeichnet
- `stop()`, wird aufgerufen, wenn der Browser das Applet anhält, (beispielsweise weil der Anwender die Webseite verlassen und zu einer anderen Webseite gewechselt hat).
- `destroy()`, wird aufgerufen, bevor der Browser das Applet aus dem Arbeitsspeicher entfernt.

Unser Applet ist jetzt zwar funktionsfähig, aber es macht noch nichts. Um es zum Leben zu erwecken, überschreiben wir die von der Basisklasse `Applet` geerbten Methoden (in diesem Abschnitt begnügen wir uns damit, die `paint()`-Methode zu überschreiben).

## Methoden überschreiben

Eine Methode zu überschreiben, bedeutet, die Signatur der Methode (Name, Parameter, Rückgabewert) zu übernehmen und mit einem neuen Anweisungsteil zu verbinden. Die `paint()`-Methode, die wir von der Basisklasse `Applet` erbt haben und die der Browser zum Zeichnen des Applets aufruft, hat beispielsweise folgende Signatur:

```
void paint(java.awt.Graphics gc)
```

Indem wir diese Methode mit einem eigenen Anweisungsteil versehen, können wir steuern, wie das Applet im Browser aussehen soll. Zuvor sollten wir aber noch ein paar Worte zu obiger Methodensignatur verlieren. (Wir haben Sie gewarnt: selbst einfachste Java-Programme konfrontieren den Programmierneuling mit einer Flut an neuen Konzepten.)

## Datentypen in Java

Java ist - im Gegensatz zu JavaScript - eine streng typisierte Sprache. Das bedeutet, dass jede Variable, jede Methode und jede Klasse, die wir verwenden wollen, vorab deklariert<sup>3</sup> werden muss und dass zu jeder Deklaration einer Variablen, eines Methodenparameters oder eines

Methodenrückgabewertes ein Datentypangabe gehört. Dabei unterscheidet Java zwischen

- elementaren Datentypen
- zusammengesetzten, definierten Datentypen

Elementare Datentypen sind beispielsweise Ganzzahlen, einzelne Zeichen, Boolesche Werte (siehe Tabelle 15.1).

Datentyp	Beschreibung	Wertebereich	Literal
boolean	Boolescher Wert	true, false	true, false
char	Zeichen, Buchstabe	Unicode-Werte	'c'
byte	ganze Zahl	-128 bis +127	12
short	ganze Zahl	-32768 bis 32767	12
int	ganze Zahl	-2.147.483.648 bis +2.147.483.647	12  0xC
long	ganze Zahl	-9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807	12L
float	Fließkommazahl	-3,40282347E+38 bis +3,40282347E+38	12.5f  1e5f
double	Fließkommazahl	-1,7976931348623157E+308 bis +1,7976931348623157E+308	12.5  1e5
void	kein Datentyp	-	-

**Tabelle 15.1: Elementare Datentypen in Java**

Variablen von elementaren Datentypen tauchen in Java-Programmen nur als Datenelemente von Klassen, als Parameter von Methoden oder als Rückgabewerte von Methoden auf.

Die zusammengesetzten Datentypen sind nichts anderes als die uns bekannten Klassen. Jede Klassendefinition definiert auch einen neuen Datentyp. Objekte einer Klasse sind daher »Variablen« vom Typ ihrer Klasse. (Wobei es keinen Unterschied macht, ob die Klasse aus der Java-Bibliothek stammt oder von Ihnen im Programm neu definiert wurde.)

Mit diesem Hintergrundwissen können wir die Signatur der paint()-Methode etwas besser verstehen:

```
void paint(java.awt.Graphics gc).
```

paint() ist eine Methode, die einen einzigen Parameter (dem wir hier den Namen gc gegeben

haben) vom Typ der Klasse `java.awt.Graphics` definiert. Ihr Rückgabotyp ist `void`, was besagt, dass die Methode keinen Wert zurückliefert.

## Die `paint()`-Methode

Zurück zu unserem Applet-Code. Dort wurde die `paint()`-Methode wie folgt überschrieben:

```
public void paint(Graphics gc)
{
 gc.drawString("Hallo World Wide Web!",100,100);
}
```

Denken Sie daran, dass die `paint()`-Methode vom Browser aufgerufen wird. Dieser übergibt ihr ein Objekt vom Typ der Klasse `java.awt.Graphics`, das den Teil der Webseite repräsentiert, den das Applet einnimmt. (Dass wir beim Überschreiben `Graphics` statt `java.awt.Graphics` schreiben können, liegt daran, dass wir weiter oben `java.awt.Graphics` importiert haben.)

Der Parameter `gc` repräsentiert jetzt für uns eine Art Leinwand, in die wir mit Hilfe der Methoden der Klasse `Graphics` zeichnen können. Alle Zeichenoperationen, die wir auf `gc` ausführen, werden sofort im Applet sichtbar.

Im obigen Beispiel begnügen wir uns damit, einen Text in das Applet zu zeichnen. Dazu dient die Methode `gc.drawString()`, die als Parameter die einzuzzeichnende Zeichenkette »Hallo World Wide Web!« erhält sowie die x- und y-Koordinaten der Stelle, an der der Text angezeigt werden soll.

## Rekapitulieren wir

Das Grundgerüst eines Java-Applets besteht aus einer Hauptklasse, die von der vordefinierten Klasse `java.applet.Applet` abgeleitet und als `public` deklariert wird.

```
import java.applet.Applet;
public class KLASSENNAME extends Applet
{
}
```

Von der Basisklasse `Applet` erbt die Hauptklasse die Methoden: `paint()`, `init()`, `start()`, `stop()` und `destroy()`. Will man etwas in den Bereich, den das Applet in der Webseite einnimmt, ausgeben, überschreibt man die geerbte `paint()`-Methode und zeichnet die Ausgabe in den `gc`-Parameter.



*Achten Sie bei allen Schlüsselwörtern und Bezeichnungen (Namen von Klassen, Packages, Methoden, Variablen) auf die Groß- und Kleinschreibung!*

# Applets kompilieren

Um dieses - und jedes andere - Applet auf Ihrem Computer zu erstellen und auszuführen, gehen Sie wie folgt vor:

1. Rufen Sie einen beliebigen Texteditor auf, mit dem Sie Ihren Quellcode als ASCII-Text abspeichern können.
2. Öffnen Sie ein neue Datei, tippen Sie Ihren Quelltext ein und speichern Sie die Datei unter den Namen der Applet-Klasse und mit der Dateierweiterung `.java`.

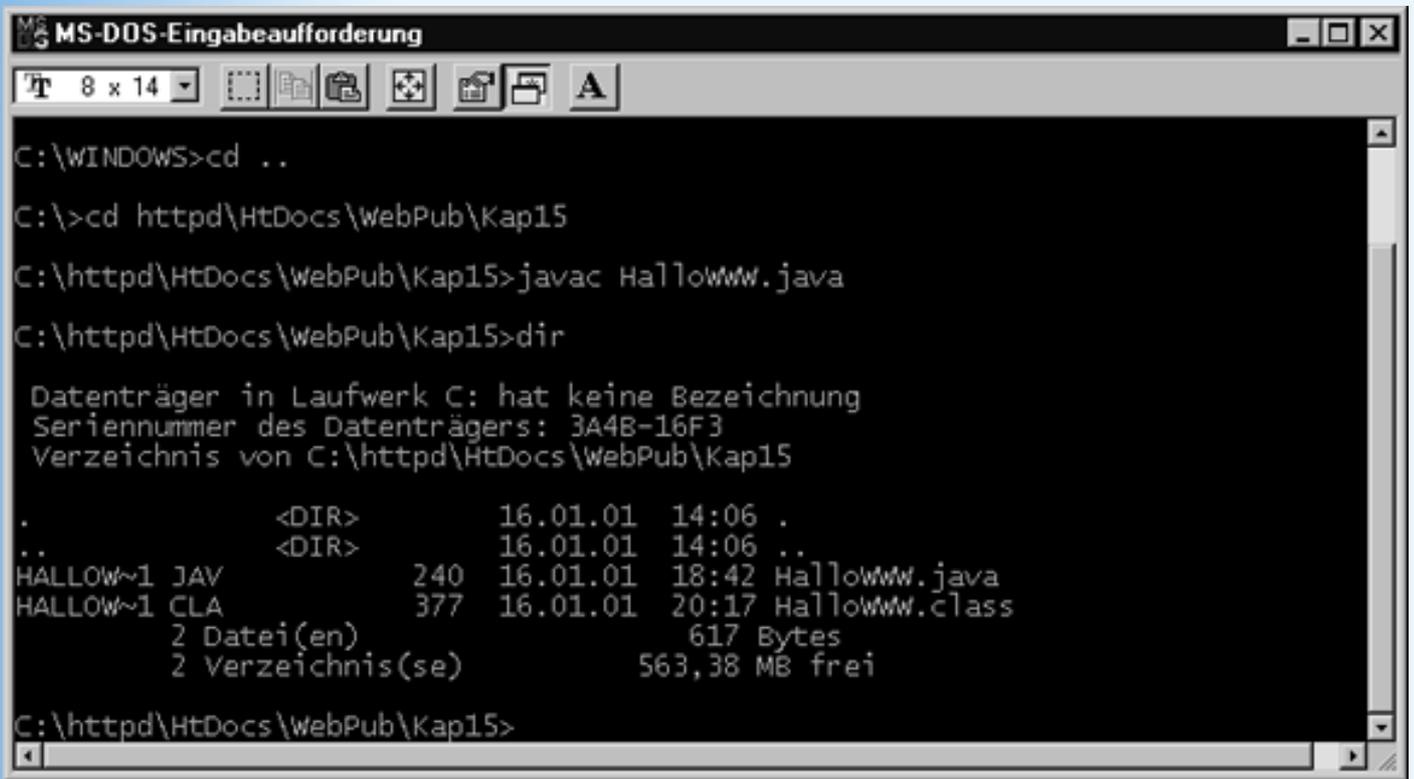


*Dies ist ganz wichtig. Wenn die Applet-Klasse, wie in unserem Beispiel `HalloWWW` heißt, muss die Datei unter den Namen `HalloWWW.java` abgespeichert werden, wobei sogar die Groß- und Kleinschreibung zu berücksichtigen ist.*

3. Öffnen Sie jetzt ein Konsolenfenster (siehe 15.2.1) und wechseln Sie zu dem Verzeichnis, in dem Sie die `.java`-Quelldatei abgespeichert haben.
4. Rufen Sie den Java-Compiler `javac` mit dem Namen der Quelldatei auf:

```
:> javac HalloWWW.java
```

Haben sich keine Tippfehler in den Code eingeschlichen, sollte in dem Verzeichnis danach die `.class`-Datei des Applets zu finden sein (`HalloWWW.class`).



```
MS-DOS-Eingabeaufforderung
8 x 14
C:\WINDOWS>cd ..
C:\>cd httpd\HtDocs\WebPub\Kap15
C:\httpd\HtDocs\WebPub\Kap15>javac HalloWWW.java
C:\httpd\HtDocs\WebPub\Kap15>dir
Datenträger in Laufwerk C: hat keine Bezeichnung
Seriennummer des Datenträgers: 3A4B-16F3
Verzeichnis von C:\httpd\HtDocs\WebPub\Kap15
. <DIR> 16.01.01 14:06 .
.. <DIR> 16.01.01 14:06 ..
HALLOW~1 JAV 240 16.01.01 18:42 HalloWWW.java
HALLOW~1 CLA 377 16.01.01 20:17 HalloWWW.class
 2 Datei(en) 617 Bytes
 2 Verzeichnis(se) 563,38 MB frei
C:\httpd\HtDocs\WebPub\Kap15>
```

Abbildung 15.4: Kompilierung des Java-Quellcodes

## Applets in Webseiten einbinden

Bevor wir das Applet ausführen und austesten können, müssen wir noch eine passende HTML-Seite aufsetzen, die das Applet aufruft.

### Listing 15.2: HalloWWW.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Erstes Applet</title>
</head>
<body>
<p>Webseite ruft Applet. Bitte antworten.</p>
<applet code="HalloWWW.class" width="300" height="100"></applet>
</body>
</html>
```

Hier wird das Applet mit Hilfe des `<applet>`-Tags eingebettet. Das ist derzeit immer noch gängige Praxis, obwohl das `<applet>`-Tag seit HTML 4 als deprecated eingestuft ist und eigentlich nicht mehr verwendet werden sollte. Statt dessen empfiehlt der Standard das `<object>`-Tag. Die Einbettung von Applets mit dem `<object>`-Tag (siehe unten), wird derzeit aber nur vom Netscape 6-Browser unterstützt. Man ist also gut beraten, im Augenblick noch dem `<applet>`-Tag den Vorzug zu geben.

Notwendige Attribute für das <applet>-Tag sind:

- code, der Name der Appletdatei (optional mit .class Extension)
- width, die Breite (in Pixeln) des Bereichs, den das Applet auf der Webseite einnimmt.
- height, die Höhe (in Pixeln) des Bereichs, den das Applet auf der Webseite einnimmt.

Wenn das Applet nicht im gleichen Verzeichnis wie das HTML-Dokument steht (genauer gesagt, nicht im aktuellen Verzeichnis), müssen Sie im codebase-Attribut den absoluten oder relativen Pfad zur Applet-Datei angeben. Wenn beispielsweise die HTML-Datei im Verzeichnis *c:\meinServer\html* steht und der Appletcode im Verzeichnis *c:\meinServer\applets*, dann wird codebase folgendermaßen gesetzt:

```
<applet code="HalloWWW.class"
 codebase=" ../applets"
 width="100" height="80">
</applet>
```

Optional kann man mit dem align-Attribut die Ausrichtung des Applets im umgebenden Text steuern:

Ausrichtung	Beschreibung
align="left"	Ausrichtung zum linken Rand hin
align="right"	Ausrichtung zum rechten Rand hin
align="top"	Ausrichtung nach dem größten Element der Zeile
align="middle"	Ausrichtung der Zeile an der Mitte des Applets
align="bottom"	Ausrichtung der Zeile am unteren Ende des Applets

**Tabelle 15.2: Ausrichtungsmöglichkeiten für Applets**

## Das <object>-Tag

Mit Hilfe des <object>-Tags sähe die Einbettung des Applets wie folgt aus:

```
<p>
<object codetype="application/java" classid="java:HalloWWW.class"
 width="300" height="100">
 Sorry, Ihr Browser unterstützt keine Java-Applets.</object>
</p>
```

Derzeit unterstützt aber wie gesagt nur der Netscape 6-Browser diese Syntax. (Dafür hapert es bei ihm mit dem korrekten Neuzeichnen der Applets - jedenfalls galt dies für das erste Release des Browsers.)

1. Speichern Sie den HTML-Code der Webseite unter einem beliebigen Namen (wir haben die HTML-Datei nach dem Applet benannt) in dem gleichen Verzeichnis wie das Java-Applet.
2. Laden Sie die HTML-Datei in Ihren Browser.



Abbildung 15.5: Das Applet im Internet Explorer

## 15.3 Parameter an Applets übergeben

Aus der Arbeit mit Funktionen und Methoden wissen Sie, dass man Funktionen oder Methoden, die Daten verarbeiten, in vielen Fällen vielseitiger einsetzen kann, wenn die Daten als Parameter übergeben werden. So wäre eine `drawString`-Methode, die immer den gleichen Text ausgibt, ziemlich nutzlos (außer man will ausgerechnet diesen einen Text ausgeben). Eine `drawString`-Methode, die einen beliebigen Text ausgibt, den man als Argument beim Aufruf übergibt, ist dagegen sehr vielseitig verwendbar.

Gleiches gilt für Applets. Aber wer könnte überhaupt den Parametern eines Applets Argumente übergeben? Richtig, die Webseite - oder genauer gesagt, der HTML-Code, der das Applet einbettet.

Parameter an Applets werden mit dem `<param>`-Tag übergeben (das zusammen mit dem `<applet>` wie auch dem `<object>`-Tag verwendet werden kann). Es können beliebig viele Parameter angegeben werden. Jeder einzelne Parameter ist durch seinen Parameternamen (`name`-Attribut) und einen Wert (`value`-Attribut) definiert.

Die Webseite aus Listing 15.3 bettet ein Applet namens `Parameter.class` ein und übergibt ihm

einen Text, den das Applet ausgeben soll.

### Listing 15.3: Parameter.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Erstes Applet</title>
</head>
<body>

<p>Webseite ruft Applet. Bitte antworten.</p>
<applet code="Parameter.class" width="300" height="100">
 <param name="text" value="Hallo vom Applet" />
</applet>
</body>
</html>
```

Jetzt müssen wir noch das dazugehörige Applet aufsetzen, das diesen Parameter einliest und verarbeitet.

### Listing 15.4: Parameter.java

```
// Parameterübergabe für Applets
import java.awt.Graphics;
import java.applet.Applet;
public class Parameter extends Applet
{
 public void paint(Graphics gc)
 {
 String str;

 str = getParameter("text");
 gc.drawString(str,100,50);
 }
}
```

Obiges Listing ist eine einfache Erweiterung des *HalloWWW.java*-Listings. Da wir die Quelldatei unter einem anderen Namen abgespeichert haben (*Parameter.java*) müssen wir auch den Namen der Hauptklasse ändern.



*Nicht vergessen! Der Name der .java-Datei und der Name der Hauptklasse des*

*Applets müssen identisch sein (einschließlich Berücksichtigung der Groß- und Kleinschreibung).*

Der eigentliche Code zur Entgegennahme und Verarbeitung des HTML-Parameters erfolgt in der `paint()`-Methode. Zuerst deklarieren wir eine String-Variable, in der wir das vom HTML-Code übergebene Argument abspeichern wollen. Java kennt keinen elementaren Datentyp für Strings (der Datentyp `char` ist nur für einzelne Zeichen). Als Ersatz gibt es in den Java-Bibliotheken eine eigene Klasse `String`, mit der man Text bearbeiten kann.

```
String str;
```

erzeugt also ein Objekt `str` vom Typ der Klasse `String`.

Als Nächstes nehmen wir den HTML-Parameter entgegen. Dazu rufen wir die Methode `getParameter()` auf. Wo kommt diese Methode her? Wir haben Sie zusammen mit den anderen Applet-Methoden (`paint()`, `init()`, `start()`, etc.) von der Basisklasse `Applet` geerbt.

Da die Methode `getParameter()` bereits in der Basisklasse so definiert wurde, dass sie einzelne Argumente vom HTML-Code entgegen nimmt und als Ergebnis zurückliefert, gibt es für uns keinen Grund diese Methode zu überschreiben (wie wir es mit `paint()` getan haben. Wir rufen Sie also einfach auf.

### Methodenaufrufe mit und ohne Objekt

Vielleicht fragen Sie sich jetzt, warum `getParameter()` einfach so aufgerufen werden kann, während wir die Methode `drawString()` über das Objekt `gc` aufrufen müssen?

Grundsätzlich gilt: Wenn Sie in einer Methode einer Klasse A eine andere Methode der Klasse A aufrufen wollen, brauchen Sie dafür kein Objekt. Sie rufen die Methode einfach auf. Wenn Sie dagegen in einer Methode einer Klasse A eine Methode einer anderen Klasse B aufrufen wollen (wie im Falle der `Graphics`-Methode `drawString()`, die wir in der Applet-Methode `paint()` aufrufen), brauchen Sie ein Objekt der anderen Klasse B.

Zurück zu unserem Applet.

Wenn wir die Methode `getParameter()` aufrufen, übergeben wir ihr als Argument den Namen des HTML-Parameters, den die Methode zurückliefern soll. In unserem Beispiel heißt der Parameter "text". Dafür liefert uns die Methode den Wert des `value`-Attributs dieses Parameters.

Wir speichern den Wert in unserem `str`-Objekt und geben dieses wie gehabt aus:

```
str = getParameter("text");
gc.drawString(str, 100, 50);
```

Nachdem Sie den Quelltext des Applets kompiliert haben, können Sie die zugehörige Webseite in Ihren Browser laden und das Applet testen.



**Abbildung 15.6: Ausgabe von Text, der als HTML-Parameter übergeben wurde**



*Die Methode `getParameter()` liefert die Werte der HTML-Parameter immer als Strings zurück. Wenn Sie Zahlen als Argumente übergeben wollen, müssen Sie den zurückgelieferten String erst in die entsprechende Zahl umwandeln. Dies geschieht mit Hilfe spezieller Methoden:*

```
String str;
int param1;
str = getParameter("Param1");
param1 = Integer.valueOf(str).intValue();
```

## Die `init()`-Methode

Obiges Beispiel hat einen kleinen Mangel. Die `paint()`-Methode wird jedes Mal aufgerufen, wenn der Browser entscheidet, dass das Applet neu gezeichnet werden muss. Dies ist zum Beispiel der Fall,

- wenn der Leser der Webseite das Applet aus dem Anzeigebereich scrollt und später wieder zu dem Teil mit dem Applet zurückkehrt
- wenn der Leser das Browserfenster verkleinert und wieder vergrößert
- wenn der Leser eine andere Anwendung öffnet, die das Browserfenster verdeckt, und später das Browserfenster wieder in den Vordergrund holt.

Unschön ist, dass dabei jedes Mal der HTML-Parameter abgefragt wird. Effizienter wäre es, den

Parameter (der sich ja nicht ändert) nur einmal, beim Laden der Webseite, abzufragen. Dazu brauchen wir nur die geerbte `init()`-Methode zu überschreiben.

### Listing 15.5: Parameter2.java

```
// Parameterübergabe für Applets
import java.awt.Graphics;
import java.applet.Applet;
public class Parameter2 extends Applet
{
 String str;
 public void init()
 {
 str = getParameter("text");
 }
 public void paint(Graphics gc)
 {
 gc.drawString(str,100,50);
 }
}
```

Hier wird der HTML-Parameter in der `init()`-Methode abgefragt und in der `paint()`-Methode ausgegeben. Damit das `str`-Objekt in beiden Methoden verfügbar ist, müssen wir es als Datenelement am Anfang der Klassendefinition deklarieren.



*Variablen, die am Anfang einer Klassendeklaration definiert werden, können von allen Methoden der Klasse benutzt werden (im Gegensatz zu lokalen Variablen, die in einer Methode definiert werden und nur in dieser Methode verwendet werden können). Man bezeichnet sie auch als Instanzvariablen.*

## 15.4 Grafikausgaben in Applets

Wenn ein Applet in eine Webseite eingefügt wird, wird ihm vom HTML-Code durch die `<applet>`-Argumente `width` und `height` eine begrenzte Fläche zur freien Verfügung gestellt. Diese Fläche ist quasi Benutzeroberfläche und Ausgabefläche unseres Applets. In der `paint()`-Methode können wir in diese Fläche zeichnen.

### Die `paint()`-Methode

Warum erzeugt man die Ausgaben eines Applets gerade in `paint()` und nicht in einer beliebigen anderen Methode des Applets?

Nun, grundsätzlich kann man tatsächlich von jeder Applet-Methode aus in das Applet zeichnen. Die `paint()`-Methode bietet aber zwei wichtige Vorteile:

- Erstens braucht man zum Zeichnen ein Objekt, das die Anzeigefläche des Applets auf der Webseite repräsentiert und auf dem man die Zeichenoperationen ausführen kann. Die `paint()`-Methode stellt uns dieses Objekt in Form des `Graphics`-Parameters `gc` automatisch zur Verfügung (wir brauchen das Objekt also nicht selbst zu erzeugen).
- Zweitens geht die gesamte Ausgabe eines Applets verloren, wenn das Browserfenster minimiert, verdeckt, das Applet aus dem Anzeigebereich gescrollt oder die Webseite verlassen wird. Holt der Websurfer das Applet nach einer solchen Aktion wieder auf den Bildschirm, muss man die Ausgabe rekonstruieren. Der Browser ruft dazu automatisch die Methode `paint()` auf. Alle Zeichenausgaben, die in `paint()` stehen, werden daher in diesen Fällen automatisch neu ausgeführt und die Ausgabe des Applets baut sich quasi von selbst wieder auf. Zeichenausgaben aus anderen Methoden werden dagegen nicht rekonstruiert.

## Die Zeichenmethoden

Alle Zeichenmethoden, die wir benötigen, sind in der Klasse `Graphics` definiert.

Methode	Beschreibung
<code>clearRect(int, int, int, int)</code>	Füllt den spezifizierten Bereich mit der Hintergrundfarbe.
<code>drawArc(int, int, int, int, int, int)</code>	Zeichnet einen Bogen ein.
<code>drawImage(Image, int, int, ImageObserver)</code>	Zeichnet ein Bild an die spezifizierte Position (= linke obere Ecke).
<code>drawLine(int, int, int, int)</code>	Zeichnet eine Linie ein.
<code>drawOval(int, int, int, int)</code>	Zeichnet ein Oval ein.
<code>drawPolygon(int[], int[], int)</code>	Zeichnet ein Polygon ein.
<code>drawPolyline(int[], int[], int)</code>	Zeichnet einen Linienzug ein.

<code>drawRect(int, int, int, int)</code>	Zeichnet ein Rechteck ein.
<code>drawRoundRect(int, int, int, int, int, int)</code>	Zeichnet ein Rechteck mit abgerundeten Ecken ein.
<code>drawString(String, int, int)</code>	Gibt einen String an der spezifizierten Koordinate aus.
<code>fillArc()</code> , <code>fillOval()</code> , <code>fillPolygon()</code> , <code>fillRect()</code> ...	Eine Reihe von Methoden zum Zeichnen ausgefüllter Formen (vgl. <code>draw...()</code> ).
<code>Color getColor()</code>	Liefert die aktuelle Zeichenfarbe zurück.
<code>Font getFont()</code>	Liefert die aktuell verwendete Schriftart zurück.
<code>setColor(Color)</code>	Definiert die beim Zeichnen zu verwendende Farbe.
<code>setFont(Font)</code>	Definiert die zu verwendende Schriftart.

**Tabelle 15.3: Auswahl einiger Methoden der Klasse Graphics**

Was kann man mit diesen Methoden anfangen? Nun, eine ganze Menge. Sie müssen nur noch wissen, dass

- Koordinaten Bildschirmpixeln entsprechen und daher ganzzahlig sind,
- der Nullpunkt des Koordinatensystems in der linken oberen Ecke liegt

und schon kann es losgehen.

## Das Flecken-Applet

Das Applet zu diesem Abschnitt ist ein einfacher Fleckengenerator, das sieben Flecke einer Farbe, aber unterschiedlicher Radien und Mittelpunkte ausgibt. Wir wollen das Applet so implementieren, dass die Farbe beim Laden des Applets und die Radien und Mittelpunkte der Flecke beim Starten festgelegt werden.

Wir beginnen damit, dass wir in der Applet-Klasse eine untergeordnete Hilfsklasse namens Scheibe definieren. Diese Klasse verfügt über Instanzvariablen, in denen man den Mittelpunkt

und den Radius eines einzelnen Flecks speichern kann.

### Listing 15.6: Spot.java

```
// Der Fleckengenerator
import java.awt.*;
import java.applet.*;
public class Spot extends Applet
{
 class Scheibe
 {
 int m_x, m_y; // Mittelpunkt
 int m_r; // Radius
 }
 Scheibe[] m_flecken; // Array von Scheibe-Objekten
 Color m_farbe;
}
```

Unter der Hilfsklasse Scheibe haben wir ein Array-Objekt namens m\_flecken erzeugt. Da das Array-Objekt mit dem Datentyp Scheibe deklariert wurde, kann man in dem Array auch nur Objekte vom Typ Scheibe abspeichern (geschieht in der init()-Methode).

Als zweite Instanzvariable der Klasse Spot haben wir m\_farbe definiert. m\_farbe ist vom Typ der vordefinierten Java-Klasse Color, und wir werden in ihr die Farbe für die Flecken speichern.

Nachdem die Instanzvariablen deklariert sind, kommen wir zur init()-Methode.

```
public void init()
{
 // Flecken erzeugen
 m_flecken = new Scheibe[7];
 for(int i = 0; i < m_flecken.length; i++)
 {
 m_flecken[i] = new Scheibe();
 }
 //Farbe festlegen
 m_farbe = new Color((int) (255*Math.random()),
 (int) (255*Math.random()),
 (int) (255*Math.random()));
}
```

Wir beginnen damit, dass wir der Instanzvariablen m\_flecken einen Wert zuweisen. m\_flecken erwartet als »Wert« ein Array-Objekt. Objekte von Klassentypen werden in Java immer mit Hilfe des new-Operators und des Konstruktors der Klasse erzeugt.

## Konstruktoren

Der Konstruktor ist eine spezielle Methode der Klasse, die den gleichen Namen wie die Klasse trägt und immer dann aufgerufen wird, wenn ein Objekt der Klasse erzeugt wird. (Tatsächlich kann der Konstruktor **nur** zur Erzeugung von Objekten aufgerufen werden.) Falls eine Klasse keinen eigenen Konstruktor definiert, weist ihr der Java-Compiler einen Standardkonstruktor zu.

Die Anweisung

```
m_flecken = new Scheibe[7];
```

erzeugt also ein Array-Objekt, das sieben Scheibe-Objekte aufnehmen kann, und weist dieses Objekt der Instanzvariablen `m_flecken` zu.

Damit haben wir aber noch nicht die einzelnen Scheibe-Objekte erzeugt. Dies geschieht in der anschließenden `for`-Schleife.



`for`- und `while`-Schleifen funktionieren in Java ganz so wie in JavaScript (siehe Kapitel 9.3).

Zum Schluss erzeugt die `init()`-Methode ein neues `Color`-Objekt und speichert es in der Instanzvariablen `m_farbe`.

Farbobjekte werden - wie in vielen Programmiersprachen üblich - als RGB-Farben definiert, das heißt man mischt die Farbe aus Rot-, Grün- und Blau-Anteilen zusammen. Jeder Farbanteil bekommt dabei einen Werte zwischen 0 (kein Farbanteil) und 255 (voller Farbanteil) zugewiesen. Die Mischung erfolgt nach dem Prinzip der Lichtfarben, das heißt, `Color(255, 255, 255)` wäre Weiß. (Für die wichtigsten Farben gibt es vordefinierte Werte: `Color.black`, `Color.red`, etc.)

Jetzt müssen wir noch festlegen, wo die Flecken eingezeichnet werden sollen und wie groß sie sein sollen. Den Code hierfür setzen wir in der `start()`-Methode auf (obwohl man ihn auch in `init()` einfügen könnte).

```
public void start()
{
 //Mittelpunkte und Radien festlegen
 for(int i = 0; i < m_flecken.length; i++)
 {
 m_flecken[i].m_x = (int) (400*Math.random());
 m_flecken[i].m_y = (int) (200*Math.random());
 m_flecken[i].m_r = (int) (50*Math.random());
 }
}
```

Zu guter Letzt geben wir die Flecken aus:

```
public void paint(Graphics gc)
{
 gc.setColor(m_farbe);
 for(int i = 0; i < m_flecken.length; i++)
 {
 gc.fillOval(m_flecken[i].m_x,
 m_flecken[i].m_y,
 m_flecken[i].m_r * 2,
 m_flecken[i].m_r * 2);
 }
}
} // Ende von Spot.java
```

Zuerst richten wir mit Hilfe der Graphics-Methode setColor() den Farbwert in m\_farbe als neue Zeichenfarbe ein.

```
gc.setColor(m_farbe);
```

Dann gehen wir in einer Schleife die einzelnen Scheibe-Objekte durch und zeichnen sie in das Applet.

Zum Zeichnen von ausgefüllten Scheiben verwendet man die Methode fillOval(). Dieser übergibt man als Argumente die Koordinaten der linken, oberen Ecke und die Breite und Höhe eines rechteckigen Ausgabebereichs. In diesen Ausgabebereich zeichnet die Methode die Scheibe formatfüllend ein. (Um kreisförmige Scheiben einzuzichnen, übergibt man einfach gleiche Werte für Breite und Höhe.)

### **Listing 15.7: Spot.html - das HTML-Dokument zum Anzeigen des Applets**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Fleckenapplet</title>
</head>
<body>
<p>Dieses Dokument dient der Demonstration des Fleckengenerators.</p>
<hr />
<applet code="Spot.class" width="400" height="200"></applet>
<hr />
</body>
</html>
```

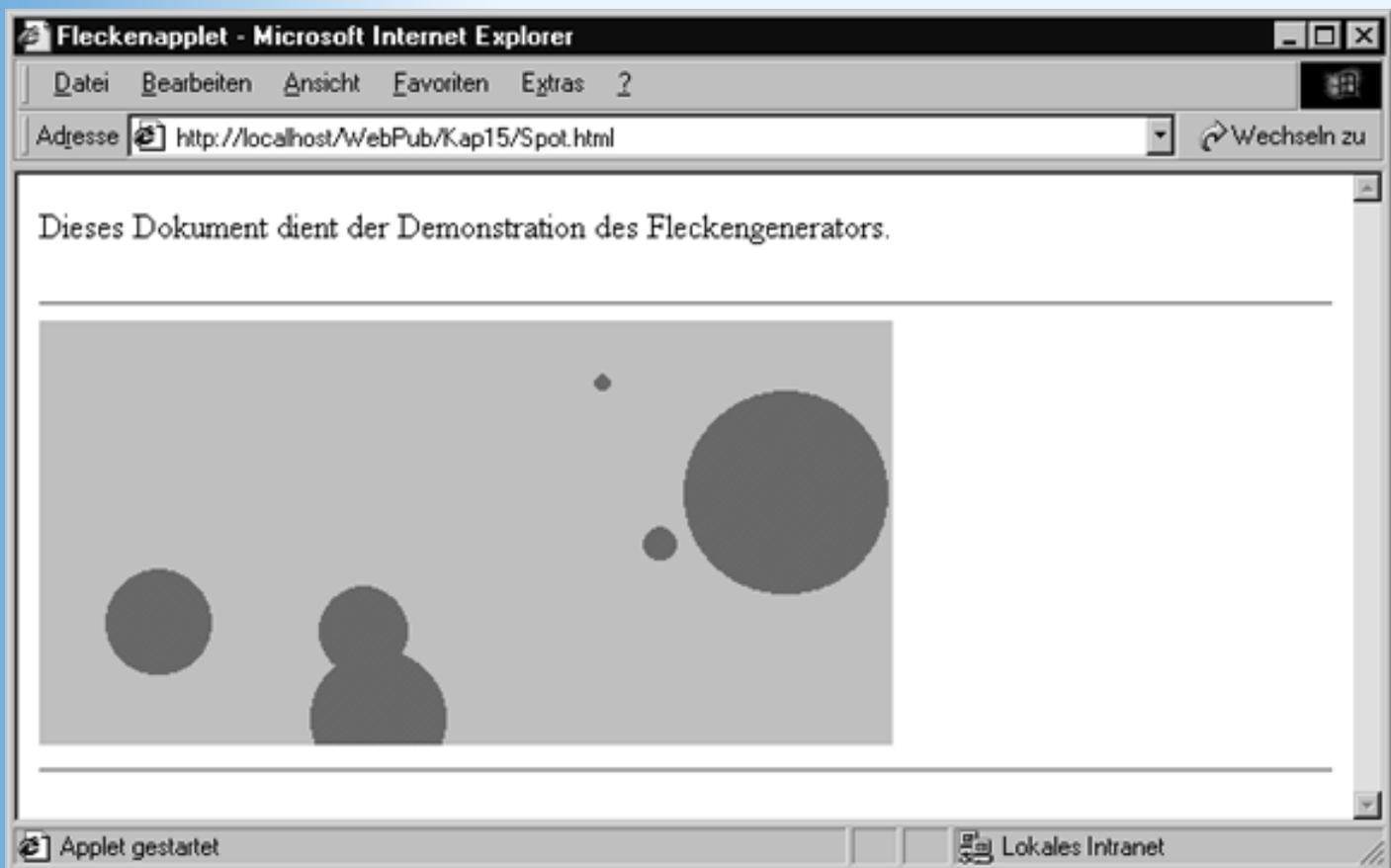


Abbildung 15.7: Das Fleckenapplet im Internet Explorer

## Das Applet testen

Nachdem Sie das Applet kompiliert haben, sollten Sie es in verschiedenen Browsern testen. Wenn Sie es beispielsweise in den Internet Explorer 5 oder den Netscape 6-Browser laden, können Sie sehen, dass bei jedem Aktualisieren (Neuladen) der Webseite die Farbe und die Positionen/Radien der Flecke neu festgelegt werden.

Wenn Sie die Webseite mit dem Applet dagegen in den Netscape Navigator 4 laden, und dann auf die Schaltfläche **Neu laden** drücken, werden Sie feststellen, dass sich nur die Position und die Größe der Flecken ändert, nicht aber die Farbe. Dies liegt daran, dass der Navigator 4 das Applet selbst nicht neu lädt, sondern nur die `start()`-Methode erneut aufruft. Soll das Applet ganz neu geladen (und `init()` aufgerufen) werden, müssen Sie den **Neu laden**-Schalter zusammen mit der (Shift)-Taste drücken.

Zum Testen können Sie die Webseite mit dem Applet auch in den Appletviewer des Java- SDK laden. Rufen Sie dazu den Appletviewer wie folgt von der Konsole aus auf:

```
prompt:> appletviewer spot.html
```

Im Appletviewer finden Sie dann eine Reihe von Schaltern, mit denen Sie das Starten, Stoppen oder Neuladen des Applets simulieren können.

## 15.5 Auf Mausklicks reagieren

Das Grundprinzip der Ereignisverarbeitung unter Java ist das gleiche wie unter JavaScript. Mit Maus und Tastatur löst der Anwender auf den verschiedenen Elementen eines Applets Ereignisse aus. Wurde für eines der ausgelösten Ereignisse eine Behandlungsmethode eingerichtet, wird der Code dieser Methode ausgeführt.

Die Frage ist nun, wie man eine solche Behandlungsmethode einrichten kann?

Zuerst überlegt man sich, für welches »Element« man welches Ereignis abfangen möchte. In Java kann man Ereignisse für das Applet oder für die im Applet enthaltenen Steuerelemente abfangen. Da wir über die Erzeugung und Einbindung von Steuerelementen in Java noch nichts gehört haben (und im Verlaufe dieses Buches auch nichts weiter erfahren werden, sorry!), werden wir uns ein Applet-Ereignis aussuchen.

Für die Applet-Programmierung sind besonders die Mausereignisse interessant (Klick mit der Maus, Maus wird über das Applet bewegt, etc.). Unter diesen Ereignissen greifen wir uns das Mausklickereignis heraus. Unser Ziel soll es sein, ein Applet zu implementieren, das Mausklicke abfängt und an der Stelle des Mausklicks eine Meldung ausgibt.

## Die Ereignismethoden

Java hält für die verschiedenen Ereignisse eine Reihe von vordefinierten Methoden bereit. Diese Methoden sind in Interfaces und Adapter-Klassen organisiert. Der Einfachheit halber werden wir uns hier auf die Adapter-Klassen konzentrieren.

Die Adapter-Klassen für die Mausereignisse heißen `MouseAdapter` und `MouseMotionAdapter`.

Adapter-Klasse	Methoden
<code>MouseAdapter</code>	<code>mouseClicked(MouseEvent)</code> <code>mouseEntered(MouseEvent)</code> <code>mouseExited(MouseEvent)</code> <code>mousePressed(MouseEvent)</code>
<code>MouseMotionAdapter</code>	<code>mouseDragged(MouseEvent)</code> <code>mouseMoved(MouseEvent)</code>

**Tabelle 15.4: Mausbehandlungsmethoden**

Das Erste, was wir tun werden, ist daher die Methode `mouseClicked` zu überschreiben. Dazu sind drei Schritte nötig.

1. Wir müssen die Adapter-Klassen in `java.awt.event` importieren.

```
import java.awt.event.*;
```

2. Wir müssen eine eigene Ereignislauscher-Klasse von MouseAdapter ableiten:

```
class MeinMausLauscher extends MouseAdapter
{
}
```

3. Wir müssen die Methode mouseClicked() überschreiben

```
class MeinMausLauscher extends MouseAdapter
{
 public void mouseClicked(MouseEvent e)
 {
 // auszuführender Code
 }
}
```

Danach brauchen wir die Ereignisbehandlungsmethode nur noch mit dem Applet zu verbinden. Hierfür verfügt das Applet über eine geerbte Methode namens addMouseListener(). Dieser übergeben wir ein Objekt unserer Mauslauscher-Klasse.

```
MeinMausLauscher obj = new MeinMausLauscher();
addMouseListener(obj);
```

Falls man das MeinMausLauscher-Objekt, das mit new MeinMausLauscher() erzeugt wird, nur dazu braucht, um es an die Methode addMouseListener() zu übergeben, kann man den obigen Code vereinfachen, indem man das Objekt direkt im Methodenaufruf erzeugt. Es wird dann automatisch an den Parameter der Methode übergeben.

```
addMouseListener(new MeinMausLauscher());
```

Den vollständigen Code für unser Applet sehen Sie in Listing 15.8

### Listing 15.8: Mausclick.java

```
// Mausclicks
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class Mausclick extends Applet
{
 String meldung;
 int x, y;
 class MeinMausLauscher extends MouseAdapter
 {
 public void mouseClicked(MouseEvent e)
```

```

 {
 meldung = "Mausklick bei (" + String.valueOf(e.getX())
 + ", " + String.valueOf(e.getY()) + ")";
 x = e.getX();
 y = e.getY();
 repaint();
 }
}
public void init()
{
 addMouseListener(new MeinMausLauscher());
 setBackground(Color.red); // roter Applet-Hintergrund
}
public void paint(Graphics gc)
{
 gc.drawString(meldung, x, y);
}
}

```

Die mouseClicked-Methode beginnt damit, dass sie die auszugebende Meldung aufbaut. Diese besteht aus einem Text (»Mausklick bei«) und den Koordinaten des Mausclicks. Die Koordinaten des Mausclicks können wir mit Hilfe der Methoden getX() und getY() vom dem MouseEvent-Parameter der mouseClicked-Methode abfragen. Leider sind die zurückgelieferten Koordinaten echte Ganzzahlen. Wir müssen die Koordinaten daher zuerst in Strings umwandeln (mit Hilfe der Methode String.valueOf()). Den vollständig aufgebauten Meldungsstring weisen wir der Instanzvariablen meldung zu.

Anschließend speichern wir die Koordinaten des Mausclicks in den Instanzvariablen x und y und fordern das Applet auf, sich selbst neu zu zeichnen.



*Rufen Sie die paint()-Methode nie direkt auf, wenn Sie möchten, dass das Applet aktualisiert wird. Rufen Sie stattdessen die Methode repaint() auf.*



**Abbildung 15.8: Mausclicks in Applet abfangen**

## 15.6 Wie geht es weiter?

Heute haben wir Sie ein wenig in die Programmierung mit Java und in die Erstellung von Applets für Webseiten eingeführt. Etliche wichtige Techniken konnten wir dabei nur kurz erläutern, etliche weitere interessante Aspekte und Möglichkeiten konnten wir nicht einmal anreißen:

- Verwendung von Steuerelementen
- Programmierung mit Bildern
- Abspielen von Sounddateien
- Verbindung zu Datenbanken
- Exception-Behandlung
- Swing
- etc.

Immerhin, Sie wissen jetzt, wie Applets funktionieren, und sollten in der Lage sein, eigene einfache Applets zu erstellen und in Ihre Webseiten einzubinden. Außerdem haben Sie Grundkenntnisse erworben, die Ihnen morgen, wenn wir eine Laufschrift als Beispiel für eine Applet-Animationen erstellen, zugute kommen werden.

Wenn Sie sich gründlicher und tiefer in die Programmierung mit Java einarbeiten wollen, besorgen Sie sich entsprechende Fachliteratur (siehe Anhang E) oder laden Sie sich das Java-Tutorial von der Sun-Website [java.sun.com](http://java.sun.com) herunter.

## 15.7 Zusammenfassung

Heute haben Sie gelernt, wie man mit Java Applets für Webseiten erstellt.

Sie haben die Website <http://java.sun.com> besucht, von der man sich stets die aktuelle Java-Entwicklungsumgebung und aktuelle Dokumentationen und Tutorien herunterladen kann.

Wir haben Sie in die Arbeit mit den Java-Entwicklungswerkzeugen vertraut gemacht, und vielleicht haben Sie dabei sogar ein bisschen mehr über die Konsole Ihres Betriebssystems erfahren.

Danach haben wir unser erstes Applet erstellt, kompiliert und mit Hilfe des <applet>-Tags in eine Testwebseite eingebunden:

```
<applet code="HalloWWW.class"
 width="100" height="80">
</applet>
```

Die Programmiersprache Java ist rein objektorientiert und daher nicht einfach zu erlernen. So mussten Sie sich heute mit einer Vielzahl von neuen Konzepten, Techniken und Syntaxformen auseinandersetzen. Vielleicht hat Sie die Sprache Java dennoch fasziniert und Sie haben - obwohl die Applets, die wir heute erstellt haben, aus didaktischen Gründen möglichst einfach und deswegen weniger praxisrelevant waren - eine Ahnung davon bekommen, wie leistungsfähig und mächtig diese Sprache ist. Vielleicht fühlen Sie sich von Java aber auch ein wenig überfordert. Dann sollten Sie nicht verzagen, sondern sich darüber freuen, dass man viele Dinge auch mit JavaScript realisieren kann.

## 15.8 Fragen und Antworten

**Frage:**

**Auf der Buch-CD ist eine Version der Java-Entwicklungsumgebung enthalten. Sollte ich trotzdem die aktuelle Version von der Java-Website herunterladen?**

*Antwort:*

*Nein. Um die Beispiele aus diesem Buch nachzuvollziehen und sich in Java einzuarbeiten, ist die auf der Buch-CD befindliche Version absolut ausreichend. Wenn Sie allerdings professionell in die Java-Programmierung einsteigen wollen, sollten Sie sich von Zeit zu Zeit auf der Java-Sun-Website umsehen, ob nicht eine aktuellere Version der Java-Entwicklungsumgebung angeboten wird und sich diese gegebenenfalls herunterladen.*

**Frage:**

**Was ist Swing?**

*Antwort:*

*Java verfügt über zwei Sammlungen von Steuerelementen: AWT-Steuerelemente und Swing-Steuerelemente. Die AWT-Steuerelemente sind so implementiert, dass Sie auf die in den Betriebssystemen der Anwender integrierten Steuerelemente zurückgreifen, die Swing-Steuerelemente tun dies nicht. Während sich das Aussehen der AWT-Steuerelemente daher immer an das Betriebssystem des Anwenders anpasst, hat man bei Verwendung der Swing-*

*Steuerelemente die Wahl, ob man das Erscheinungsbild anpassen lassen oder fest vorgeben möchte.*

Swing ist noch relativ neu, aber es ist abzusehen, dass es sich bald durchsetzen und zumindest die alten AWT-Steuerelemente verdrängen wird. Wenn Sie nach Literatur zu Java Ausschau halten, achten Sie daher unbedingt darauf, dass in den Büchern Swing berücksichtigt wird.

## 15.9 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

1. Kann man mit Java nur Applets erstellen?
2. Wieso können Applets auf verschiedenen Plattformen ausgeführt werden?
3. Welche drei Attribute muss man angeben, wenn man ein Applet mit Hilfe des <applet>-Tags in eine Webseite einbettet?
4. Von welcher Basisklasse muss man die Hauptklasse eines Applets ableiten?
5. Wie gibt man in einem Applet Text aus?
6. Wie setzt man die Hintergrundfarbe eines Applets?

### Übungen

1. Erweitern Sie das Fleckenapplet aus Abschnitt 15.4.3 um eine Ereignisbehandlung für Mausklicks im Applet. Bei jedem Mausklick soll die Farbe der Flecken geändert werden.

---

❖ Kapitel Inhalt Index **SAMS** ❖ Top Kapitel ❖

---

1

Applet ist ein Kunstwort für »little Application« (kleine Anwendung)

2

Für eine Einführung in Klassen, Objekte und objektorientierte Grundideen siehe Kapitel 9.5).

3

»Deklarieren« bedeutet hierbei, dass die betreffenden Elemente dem

Compiler bekannt gemacht werden.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

## Tag 16

# Animationen

Die Erstellung von Animationen ist zweifelsohne eine der faszinierendsten Aufgaben eines Webdesigners. Grund genug, diesem Themen ein eigenes Kapitel zu widmen und Sie mit verschiedenen Techniken zur Erstellung von Animationen bekannt zu machen.

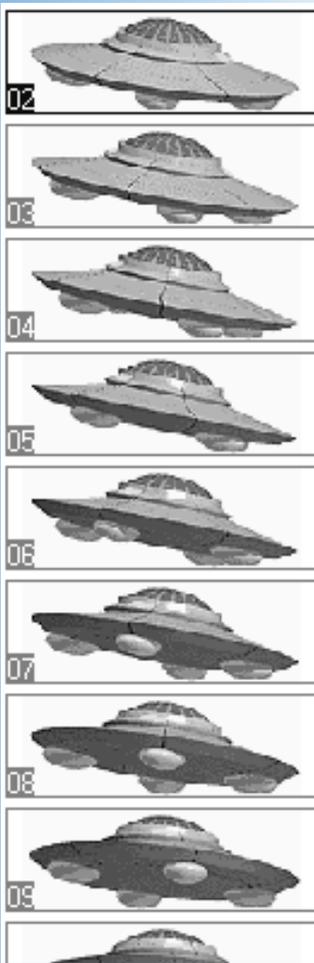
Die Themen heute

- Was sind Animationen?
- Wie erstellt man GIF-Animationen?
- Wie erstellt man browserunabhängige JavaScript-Animationen?
- Wie programmiert man Java-Animationen?
- Wie programmiert man Laufschriften?
- Wie erstellt man Flash-Animationen?
- Wie bindet man Flash-Animationen in Webseiten ein?

## 16.1 Animationen

Die Grundform der Animation ist eine Folge von Bildern, die nacheinander angezeigt werden.

- Werden ganz unterschiedliche Bilder in größeren Zeitabständen eingeblendet, erhält man eine Bildwechsler (oder eine Diashow).
- Wenn sich die Bilder (man spricht auch von Frames) nur wenig unterscheiden und wie bei einem Zeichentrickfilm relativ schnell aufeinander folgen, entsteht der Eindruck einer durchgehenden Bewegung (siehe Abbildung 16.1).



**Abbildung 16.1: Einzelne Bilder (Frames) einer Animation**

Für die Erstellung von Animationen mit bewegten Elementen gibt es wieder unterschiedliche Techniken:

- Eine Möglichkeit ist, das gesamte Bild von Frame zu Frame jeweils ein wenig zu verändern (wie in Abbildung 16.1).
- Eine andere Technik ist die Sprite-Animation, bei der einzelne Figuren oder Teile (die Sprites) vor einem fixen Hintergrund verschoben werden. Wichtig ist dabei, dass der Hintergrund der Sprites (die selbst ja auch nur rechteckige Bilder sind) transparent ist, so dass das Hintergrundbild durchscheinen kann.

In Webseiten kann man solche Animationen auf verschiedene Weisen einbinden:

- Man kann die Animation als GIF-Animation erstellen und speichern und dann wie ein ganz normales Bild als <img>-Element einbinden.
- Man kann die einzelnen Bilder mit Hilfe von JavaScript laden und abspielen.
- Man kann die Animationen mit Hilfe eines leistungsfähigen Animationsprogramms (beispielsweise Flash) erstellen und als multimediales Objekt einbetten.

Darüber hinaus erlaubt uns DHTML auch noch andere Formen der Animation, die darauf beruhen, dass man absolut positionierte HTML-Elemente via CSS-Positionierung und JavaScript verschieben kann. So kann man

- einzelne Bilder animieren, indem man sie über die Webseite bewegt

- Text oder beliebige andere HTML-Elemente animieren, über die Webseite bewegen oder aus- und einblenden.
- Bilder zur gleichen Zeit durch Wechsel der Frames und durch Verschiebung auf der Seite animieren.

Die folgenden Abschnitte sollen Ihnen einen kleinen Überblick geben und Sie in die wichtigsten Animationstechniken einführen.

## 16.2 GIF-Animationen

GIF-Animationen sind Frame-Animationen, das heißt, eine Folge von Bildern wird Bild für Bild hintereinander abgespielt. Um GIF-Animationen erstellen zu können, braucht man

- die einzelnen Bilder
- ein Programm, das die einzelnen Bilder zu einer GIF-Animation zusammenfügt. Wenn das Programm die erzeugte Animation auch noch optimiert (die Dateigröße reduziert), kann man mit solchen GIFs nette filmähnliche Animationen erstellen.

### Bildwechsler und einfache Animationen mit ImageMagick

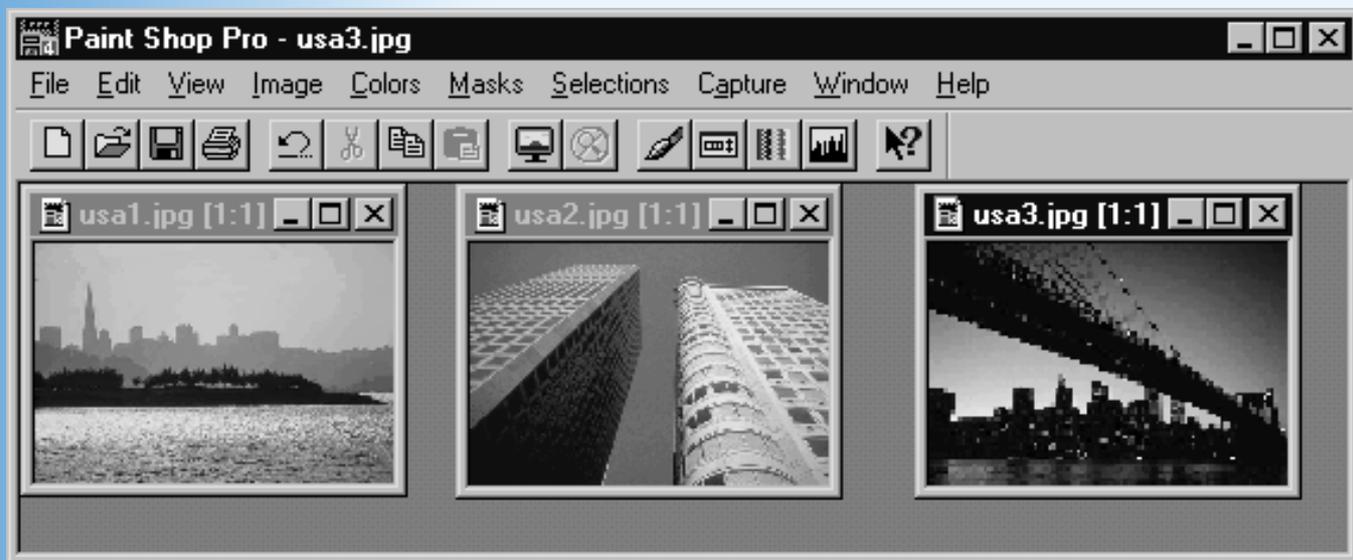
ImageMagick ist eine Sammlung von Konsolenprogrammen zur Grafikbearbeitung. Wegen der fehlenden grafischen Benutzeroberfläche ist die Bedienung etwas gewöhnungsbedürftig, aber dafür stehen die Programme kostenfrei für fast alle Plattformen zur Verfügung.



*Die ImageMagick-Programmsammlung kann von [www.simplesystems.org/ImageMagick](http://www.simplesystems.org/ImageMagick) heruntergeladen werden.*

Nachdem Sie ImageMagick heruntergeladen und installiert haben (siehe begleitende Dokumentation), brauchen Sie nur noch in einem geeigneten Grafikprogramm die Bilder für die Frames zu erstellen, und schon kann es losgehen.

Für das folgende Beispiel verwenden wir drei Photos mit Ansichten amerikanischer Städte (siehe Abbildung 16.2).



**Abbildung 16.2: Einzelne Frames für den Bilderwechsler**

Die einzelnen Frames haben alle den gleichen Namen, plus einer Indexnummer. Für die Arbeit mit dem ImageMagick-Tool ist dies ganz wichtig. Um aus diesen drei Frames eine GIF-Animation zu machen, gehen Sie wie folgt vor:

1. Kopieren Sie die einzelnen Bilder in das Verzeichnis, in dem die ImageMagick-Tools (EXE-Dateien) installiert wurden.
2. Rufen Sie eine Konsole auf (unter Windows **Start/Programme/MSDOS-Eingabeaufforderung**) und wechseln Sie in dieser in das Installationsverzeichnis von ImageMagick.
3. Rufen Sie das *convert*-Tool wie folgt auf:

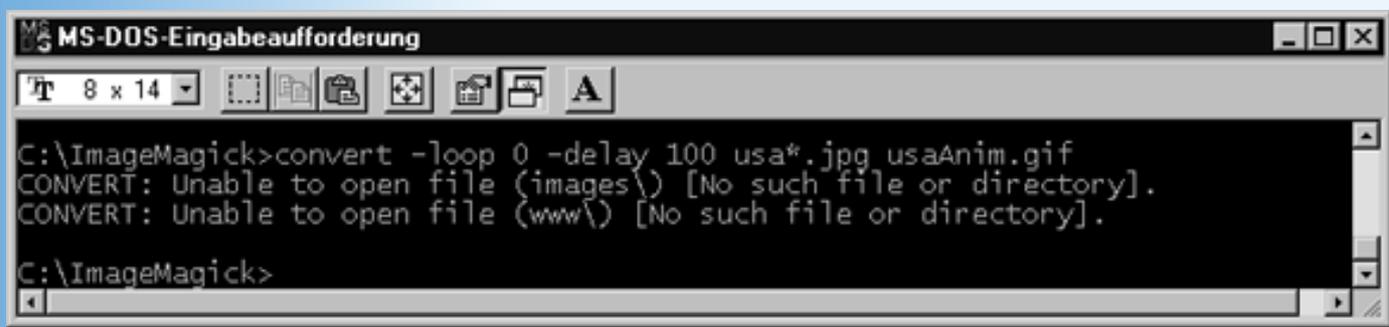
```
> convert -loop 0 -delay 100 usa*.jpg usaAnim.gif
```

Das erste Argument, `-loop 0`, gibt an, wie oft die Animation wiederholt werden soll. Der Wert 0 steht hierbei für eine endlose Wiederholung.

Der zweite Argument, `-delay 100`, gibt an, wie viele Hundertstelsekunden jedes Bild angezeigt werden soll.

Das dritte Argument gibt die Frames an. `usa*.jpg` bewirkt, dass alle JPG-Bilder, deren Dateinamen mit `usa` anfängt und die im aktuellen Verzeichnis stehen, aufgenommen werden. Die Reihenfolge der Bilder in der Animation richtet sich nach der lexikographischen Reihenfolge ihrer Dateinamen (in unserem Beispiel also nach den auf das Präfix `usa` folgenden Nummern).

Schließlich geben wir noch den Namen der zu erzeugenden Animationsdatei an (hier `usaAnim.gif`).



```
MS-DOS-Eingabeaufforderung
T 8 x 14
C:\ImageMagick>convert -loop 0 -delay 100 usa*.jpg usaAnim.gif
CONVERT: Unable to open file (images\) [No such file or directory].
CONVERT: Unable to open file (www\) [No such file or directory].
C:\ImageMagick>
```

**Abbildung 16.3: Aufruf von convert**

Die erzeugte GIF-Datei können wir in ein Verzeichnis auf dem Webserver kopieren und per `<img>`-Tag in eine Webseite einbinden:

```

```



*Auf die gleiche Weise kann man mit Hilfe des convert-Tools zeichentrickähnliche Animationen, Übergänge oder Ähnliches erstellen.*

## Animationen und Sprites mit dem Ulead GIF-Animator

Wer unter Windows arbeitet, kann zwischen einer Vielzahl von Shareware-Programmen zur Erstellung von GIF-Animationen wählen. Eines der populärsten und leistungsfähigsten ist der Ulead GIF-Animator. Von [www.ulead.com](http://www.ulead.com) können Sie sich eine Probeversion (Download/Trial-Link) herunterladen.

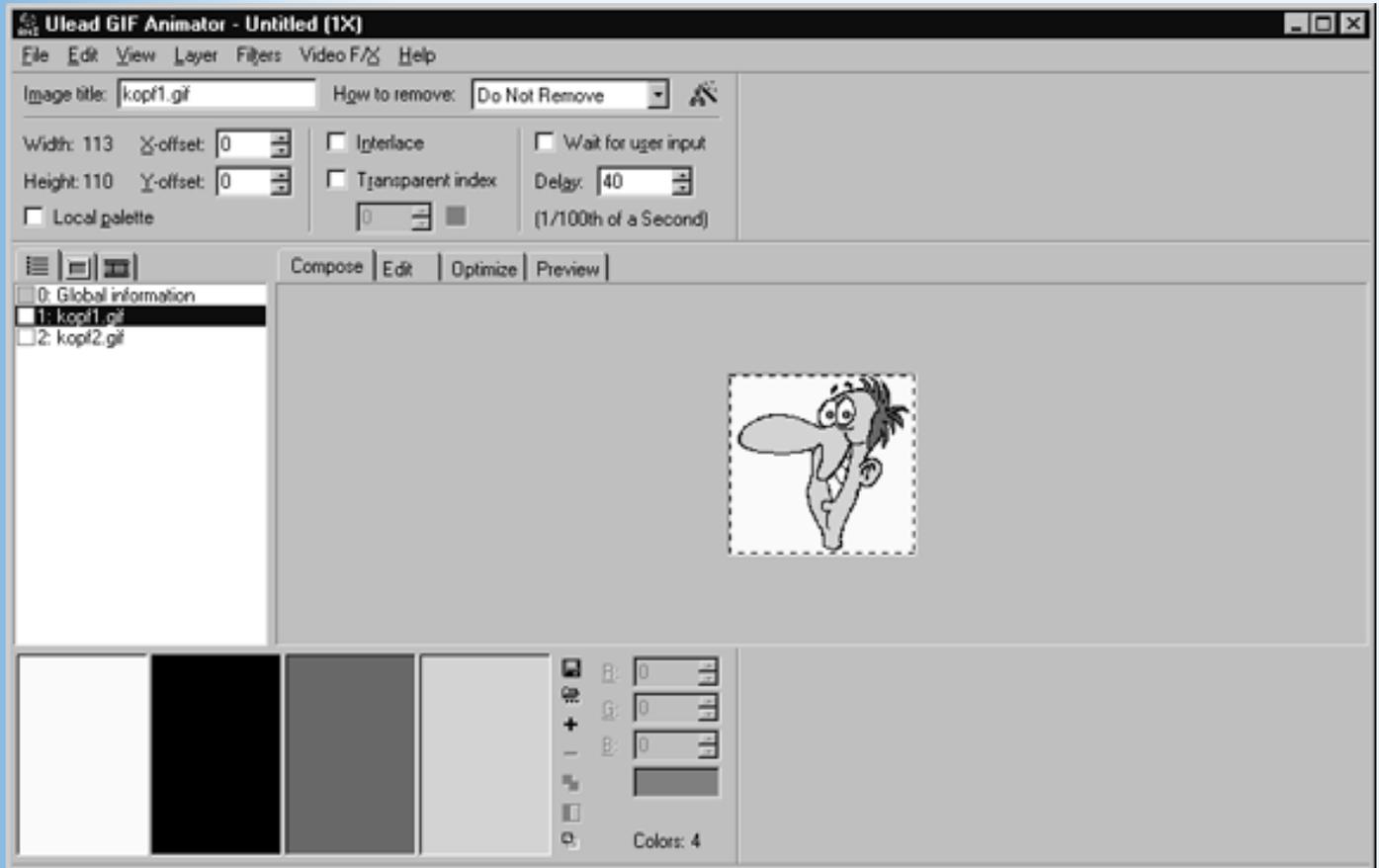
### Einfache Frame für Frame-Animationen

Sofern Sie die einzelnen Frames für die Animation bereits erstellt haben, ist die Erzeugung der GIF-Animationsdatei mit dem Ulead GIF-Animator kein Problem mehr.

1. Starten Sie die Software.
2. Beginnen Sie eine neue Animation mit dem Befehl **File/New/Document**.
3. Laden Sie Bilder mit Hilfe des Menübefehls **Layer/Add Images**.
4. Korrigieren Sie gegebenenfalls die Reihenfolge der Bilder, indem Sie den Frame eines Bildes im linken Arbeitsbereich anklicken und den markierten Frame dann mit Hilfe der Befehle **Layer/Move Layer Up** und **Layer/Move Layer Down** verschieben. (Ulead spricht von Layers statt von Frames.)
5. Legen Sie über die Eigenschaft **delay** die Anzeigedauer für die einzelnen Bilder fest (in Hundertstelsekunden). (Wenn die Werkzeugleiste mit den Bildeigenschaften nicht auf Ihrem Bildschirm angezeigt wird, rufen Sie den Befehl **View/Attribute Toolbar** auf).
6. Legen Sie die Häufigkeit der Wiederholungen fest. Klicken Sie im linken Arbeitsbereich auf den Eintrag **Global Information** und setzen Sie dann in der Attribute-Werkzeugleiste die

Eigenschaft **infinite** (endlos) beziehungsweise geben Sie darunter die Anzahl der Wiederholungen an.

- Optimieren Sie die Frames. Wählen Sie im rechten Arbeitsbereich die Registerkarte **Optimize** und klicken Sie auf **Optimize now**.
- Speichern Sie die GIF-Datei.

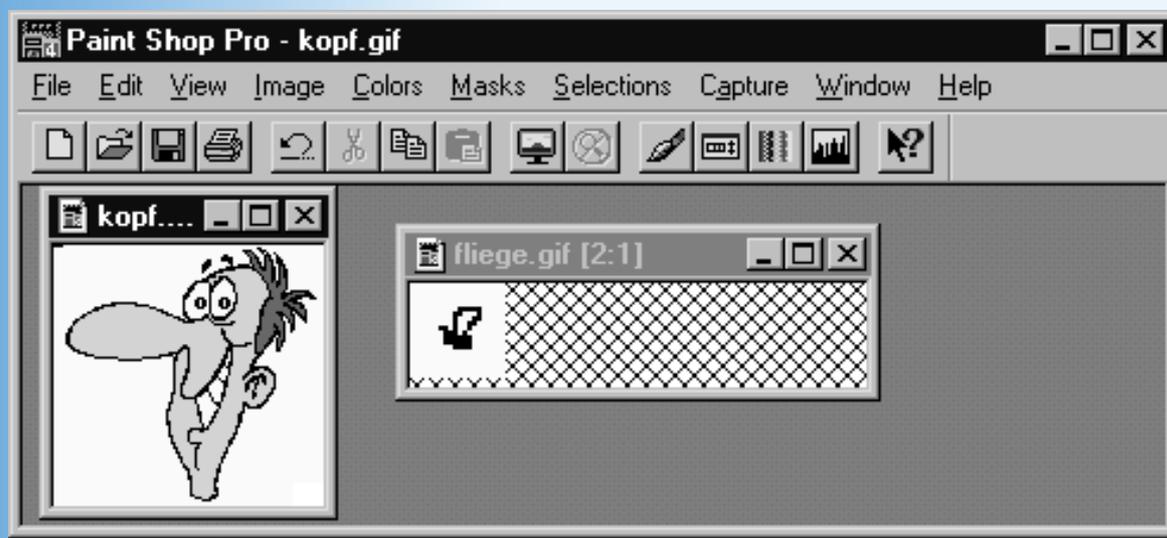


**Abbildung 16.4: Bearbeitungsstand nach Schritt 5**

Mit Ulead können Sie aber auch Animationen mit Sprites erzeugen.

## Animationen mit Sprites

Als Ausgangsmaterial für die folgende Animation dient uns die Kopf-Clipart aus dem vorangehenden Abschnitt und eine kleine selbstgezeichnete Fliege. In der GIF-Animation soll die Fliege um den Kopf herumfliegen.



**Abbildung 16.5: Die Ausgangsbilder**

Beide Bilder hätte man übrigens auch im Ulead GIF-Animator erstellen können, aber so können wir Ihnen die Bilder auf der Buch-CD zur Verfügung stellen.

## Bilder laden

1. Beginnen Sie eine neue Animation mit dem Befehl **File/New/Document**.
2. Laden Sie mit Hilfe des Menübefehls **Layer/Add Images** zuerst das Bild *kopf.gif* und dann das Bild *fliege.gif*.

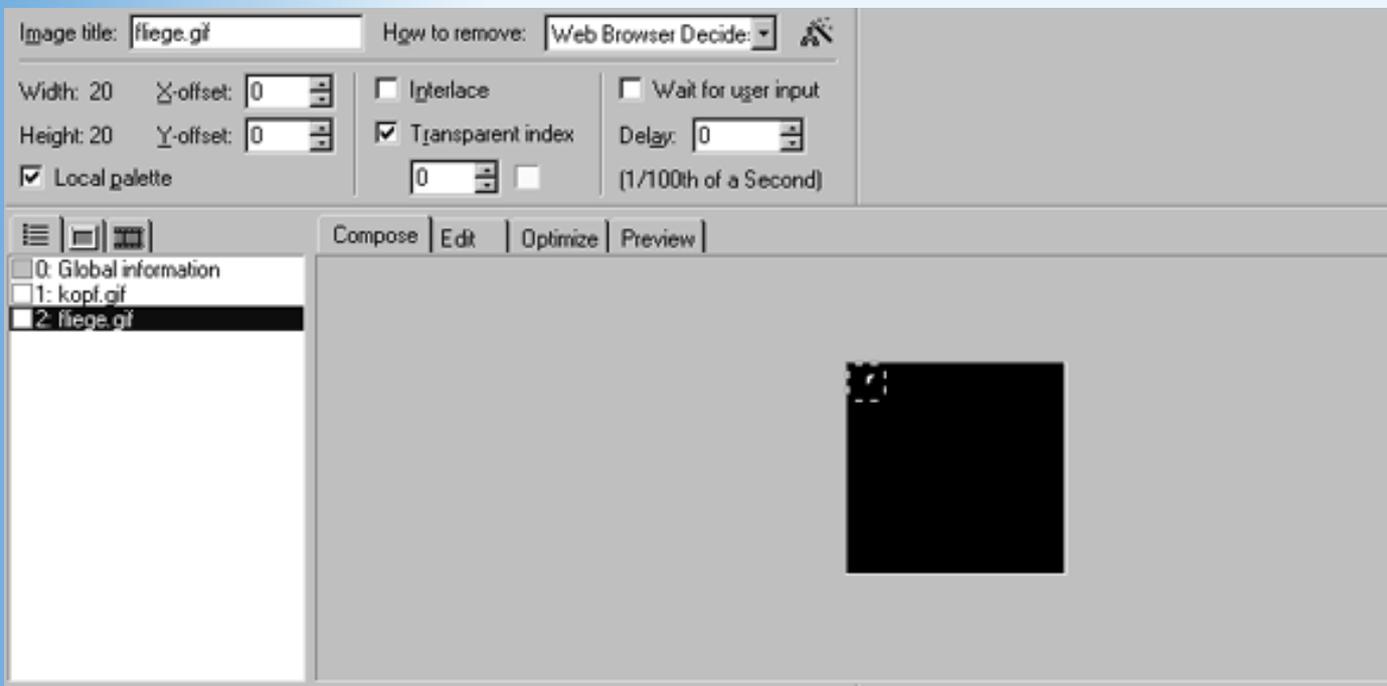
Wichtig ist, dass der Frame mit dem größten Bild zuerst kommt. Dann interpretiert Ulead die kleineren Bilder nämlich als Sprites, die über das größere Bild bewegt werden.

Korrigieren Sie gegebenenfalls die Reihenfolge der Bilder, indem Sie den Frame eines Bildes im linken Arbeitsbereich anklicken und den markierten Frame dann mit Hilfe der Befehle **Layer/Move Layer Up** und **Layer/Move Layer Down** verschieben.

Wenn Sie einen Frame löschen wollen, markieren Sie ihn im linken Arbeitsbereichfenster und rufen Sie den Menübefehl **Edit/Delete** auf.

## Bildeigenschaften kontrollieren und ändern

Nachdem Sie *fliege.gif* geladen haben, werden Sie unter Umständen enttäuscht sein, dass man die Fliege gar nicht erkennen kann, da der transparente Hintergrund der Fliege durch den globalen schwarzen Hintergrund ersetzt wird.



**Abbildung 16.6: Bildhintergrund und globaler Hintergrund passen nicht**

3. Klicken Sie dann links auf den Eintrag **Global information** und wählen Sie in der Attribute-Werkzeugleiste als Hintergrundfarbe (**background color**) den Index der weißen Farbe.



*Sie können die Hintergrundfarbe auch über die Palette auswählen. Falls die Farbpalette nicht angezeigt wird, rufen Sie zuerst den Befehl **View/Color Toolbar** auf. Klicken Sie dann auf das Farbfeld neben dem Index-Feld für die Hintergrundfarbe. Wenn Sie jetzt die Maus über die Farbpalette bewegen, verwandelt sich der Mauszeiger in eine Pipette und Sie können eine Farbe durch Klick mit der Pipette auswählen.*

Klicken Sie dann auf den Eintrag des kopf-Frames. Dieser verwendet per Voreinstellung keine transparente Farbe. Wenn Sie wollen, können Sie dem Frame aber über die Option **Transparent index** eine transparente Farbe zuweisen.

Klicken Sie zum Schluss noch einmal auf den fliege-Frame. Die Fliege sollte dank der Änderung der globalen Hintergrundfarbe jetzt gut zu erkennen sein.

## **Vorschau und Platzierung von Sprites**

Sie haben immer noch das fliege-Frame markiert. Klicken Sie jetzt im rechten Arbeitsbereich auf den Reiter **Preview**. Ulead spielt die Animation auf dem derzeitigen Arbeitsstand ab, das heißt, Sie müssten jetzt den Kopf mit einer blinkenden Fliege über der Nasenspitze sehen.



*Beachten Sie, dass die Frames nicht mehr im Wechsel eingeblendet werden. Dies liegt daran, dass die Frames unterschiedliche Bildgrößen haben. Der erste Frame wird daher als Hintergrund verwendet.*

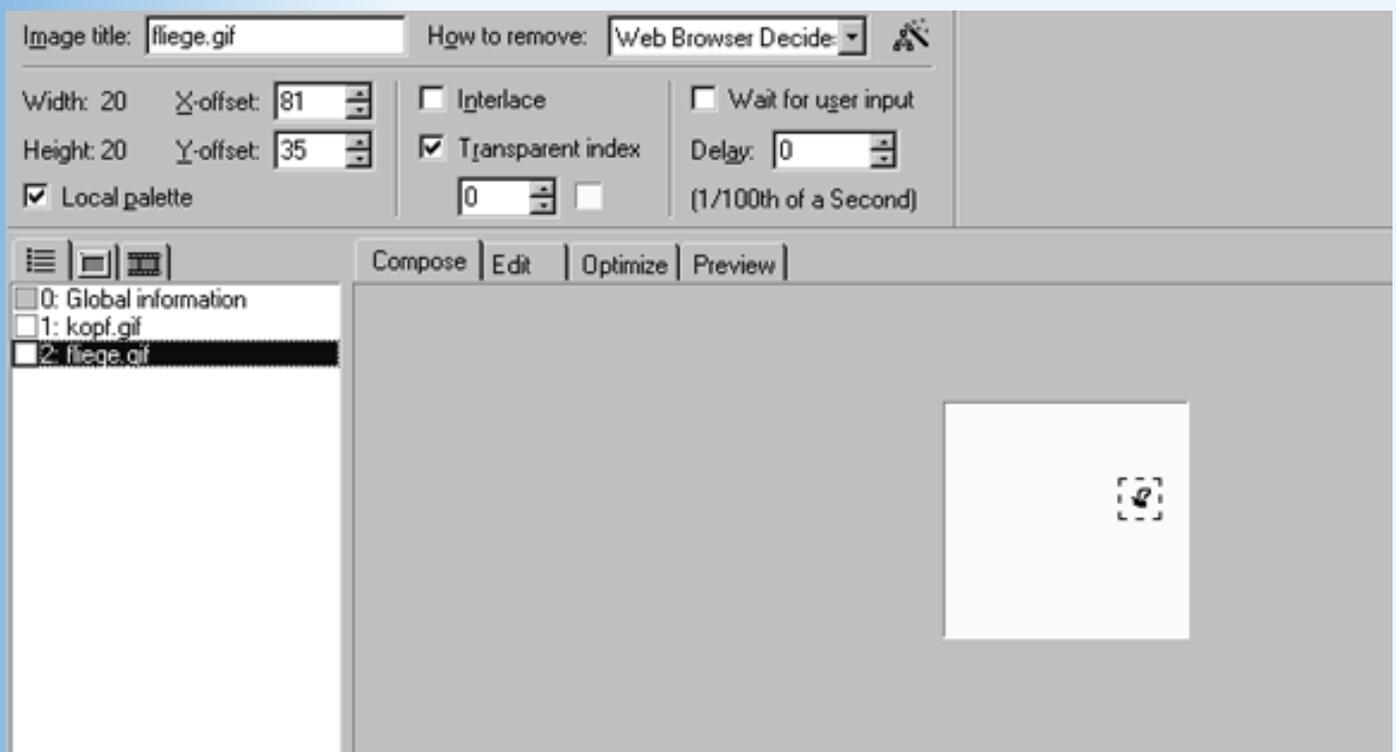
Die Fliege soll vom linken Ohr (im Bild rechts) zum unteren Teil der Nase fliegen. Wir müssen die Fliege also auf ihre Ausgangsposition am linken Ohr verschieben.

4. Wechseln Sie zum Register **Compose**. Klicken Sie dann auf das Bild der Fliege und verschieben Sie es mit gedrückt gehaltener Maustaste zu der Position, an der ungefähr das linke Ohr des Kopfes liegt.

Sie können die Position kontrollieren, indem Sie zwischendurch zur **Preview**-Ansicht wechseln.



*Alternativ können Sie die Position auch über die Felder **X-Offset** und **Y-Offset** aus der Attribute-Werkzeugleiste setzen. Gute Werte sind **X-Offset** gleich 81 und **Y-Offset** gleich 35.*



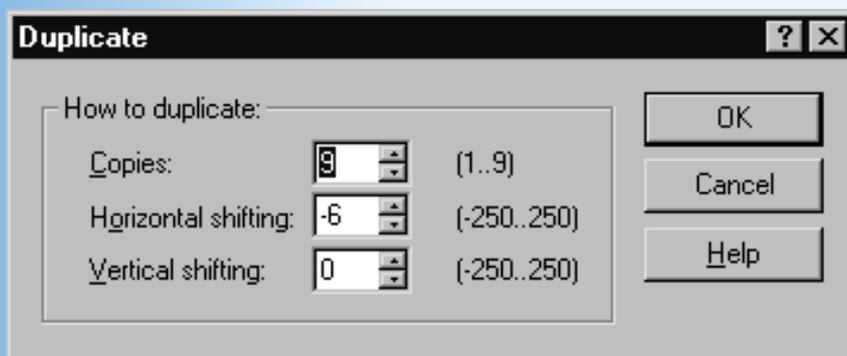
**Abbildung 16.7: Die platzierte Fliege**

## Frames kopieren und Sprites bewegen

Als Nächstes wollen wir neun Kopien des Fliegen-Frames anlegen. In jedem dieser Frames soll die Fliege ein wenig weiter nach links geflogen sein, bis sie in der neunten Kopie unter der Nase angekommen ist.

Eine Möglichkeit, die neuen Frames zu erstellen, wäre, den fliege-Frame zu markieren, dann über die Zwischenablage (Befehle **Edit/Copy** und **Edit/Paste Image**) zu kopieren, neun Mal einzufügen und die Fliege in jedem Frame neu zu positionieren. Man kann sich diese Arbeit aber auch ein wenig erleichtern.

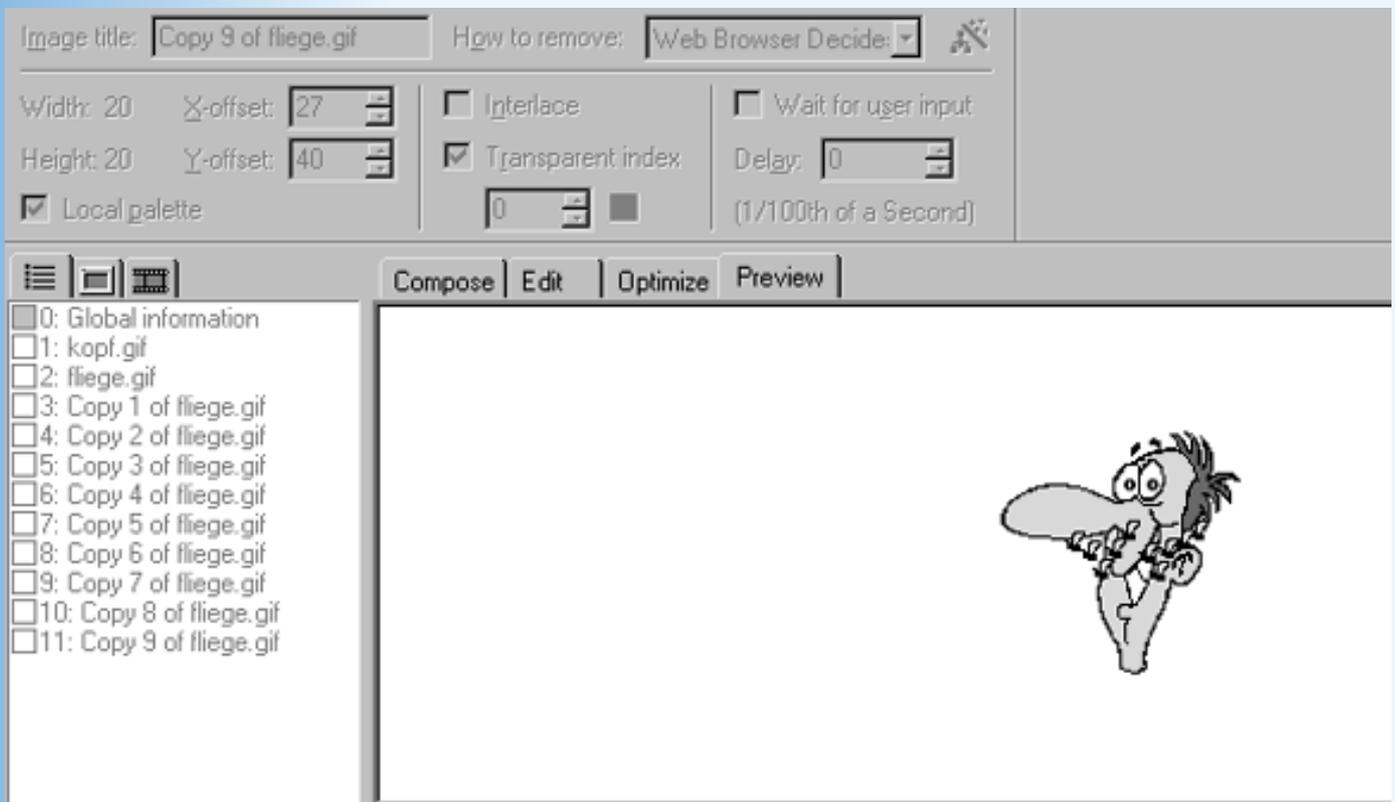
5. Markieren Sie den fliege-Frame und rufen Sie den Befehl **Edit/Duplicate** auf. In dem erscheinenden Dialogfenster geben Sie die Anzahl der anzufertigenden Kopien (9) und die Verschiebung des Sprites pro Kopie ein (wir lassen hier nur um 6 Pixel nach links verschieben).



**Abbildung 16.8: Sprite kopieren und verschieben**

6. Gehen Sie die einzelnen Frames dann durch und ziehen Sie die Fliege jeweils ein wenig mehr oder weniger nach unten, so dass sie in einer Wellenbewegung vom Ohr zur Nase fliegt. Im letzten Frame soll die Fliege halb über der Nase liegen.

Das Ergebnis können Sie in der Preview-Ansicht kontrollieren.



**Abbildung 16.9: Die Animation nach dem Kopieren**

So wie die Animation jetzt konfiguriert ist, wird die Fliege auf jeder Position eingezeichnet und bleibt eingezeichnet. Damit die vorangehenden Positionen automatisch gelöscht werden, könnte man die fliegen-Frames zusammen markieren und ihnen im Feld **How to remove** die Option **To Previous State** zuweisen. Doch leider wird diese Option nicht von allen Browsern unterstützt. Besser ist es, wenn wir die einzelnen Sprites wieder mit dem Hintergrundbild zu normalen Frames verwandeln.

## Bilder/Frames verschmelzen

7. Zuerst fertigen Sie mit Hilfe des Befehls **Edit/Duplicate** 9 Kopien des kopf-Frames an (natürlich ohne Verschiebung).
8. Dann rufen Sie den Befehl **Layer/Merge Images** auf und wählen im erscheinenden Dialogfeld den ersten kopf- und den ersten fliegen-Frame aus ((Strg)-Taste beim Anklicken gedrückt halten). Aktivieren Sie die Option **Replace original image layers**, so dass die markierten Frames nach dem Verschmelzen gelöscht werden. Drücken Sie auf OK.
9. Verfahren Sie analog mit allen neun kopf/fliege-Pärchen.

## In Frames zeichnen

Den letzten Frame, in dem die Fliege halb über der Nase liegt, müssen wir von Hand ein wenig nachkorrigieren.

10. Markieren Sie den Frame (sollte jetzt **10: Merged Layer** heißen) und wechseln Sie zur **Edit**-Registerkarte.

Vergrößern Sie das Bild gegebenenfalls durch Klick auf das Lupen-Symbol in

der Werkzeugleiste.

Klicken Sie dann in das vordere Farbfeld unter der Werkzeugleiste. Dieses Farbfeld gibt die Vordergrundfarbe beim Zeichnen an. Nachdem Sie das Feld angeklickt haben, bewegen Sie die Maus über die Nase. Der Mauszeiger nimmt die Gestalt einer Pipette an. Klicken Sie jetzt in die Nase. Die Farbe der Nase wird als neue Vordergrundzeichenfarbe ausgewählt.

Klicken Sie jetzt in der Werkzeugleiste auf den Zeichenstift. Übermalen Sie den Teil der Fliege, der über die Nase ragt.

Jetzt sollte das Bild so aussehen, als würde die Fliege unter der Nase wegfliegen.

## Anzeigedauer für die Bilder festlegen

In der Attribute-Werkzeugleiste können Sie über die **delay**-Eigenschaft festlegen, wie lange der ausgewählte Frame angezeigt werden soll (in Hunderstel Sekunden). Sollen mehrere Frames die gleiche **delay**-Eigenschaft zugewiesen bekommen, kann man die Frames gemeinsam markieren (durch Gedrückthalten der (Strg)-Taste) und dann die **delay**-Eigenschaft einmal für alle markierten Frames einstellen.

11. Setzen Sie die **delay**-Eigenschaft für die einzelnen Bilder auf 0.4 Sekunden (40 Hunderstel).

## Abschlussarbeiten

Zum Abschluss der Animation brauchen wir noch einen Kopf ohne Fliege, der den Teil der Animation darstellt, in dem die Fliege hinter dem Kopf zurück zum Ohr fliegt.

12. Markieren Sie den letzten Frame und fügen Sie dann mit Hilfe des Menübefehls **Layer/Add Images** das Bild *kopf.gif* am Ende der Frames-Liste ein.

Setzen Sie die Verweildauer für diesen Frame auf 1 Sekunden (100).

13. Legen Sie die Häufigkeit der Wiederholungen fest. Klicken Sie im linken Arbeitsbereich auf den Eintrag **Global Information** und setzen Sie dann in der Attribute-Werkzeugleiste die Eigenschaft **infinite** (endlos) beziehungsweise geben Sie darunter die Anzahl der Wiederholungen an.
14. Optimieren Sie die Frames. Wählen Sie im rechten Arbeitsbereich die Registerkarte **Optimize** und klicken Sie auf **Optimize now**.
15. Speichern Sie die GIF-Datei.
16. Setzen Sie eine Webseite auf, in der Sie die GIF-Animation einbinden. Testen Sie die Webseite in den verschiedenen Browsern.

## Listing 16.1: fliegenAnim.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
```

```
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>GIF-Animation</title>
</head>
<body>
<h1>Mit Sprite erstellte Animation</h1>
<div style="margin-left: 100px">

</div>
</body>
</html>
```

## Weitere Quellen

Im Internet gibt es eine Vielzahl von Quellen für GIF-Animations-Software oder auch fertige und frei verfügbare GIF-Animationen. Starten Sie einfach Ihren Browser, steuern Sie eine Suchseite an (yahoo, lycos, fireball, etc.) und suchen Sie nach Begriffen wie GIF oder GIF-Animation.

## 16.3 JavaScript-DHTML-Animationen

Von DHTML, oder »dynamischem HTML«, sprechen wir, wenn wir mit JavaScript auf den HTML-Code einer Webseite zugreifen und so das Erscheinungsbild der Webseite beziehungsweise einzelner HTML-Elemente verändern. Von DHTML im engeren Sinne sprechen wir, wenn wir JavaScript mit HTML/CSS-Techniken zur Positionierung und Sichtbarkeitseinstellung von HTML-Elementen kombinieren.

Ein erstes Beispiel dazu haben Sie bereits in Abschnitt 10.7 bei der Implementierung dynamischer Listen gesehen. In diesem Abschnitt wollen wir Ihnen am Beispiel einer kleinen Hai-Animation weitere DHTML-Techniken vorstellen.



*Obwohl wir in diesem Abschnitt eine zeichentrickähnliche Animation mit Bildern erstellen, ist die Animation mit Bildern keineswegs das einzige Einsatzgebiet für DHTML. Mit DHTML können Sie praktisch jedes beliebige Element einer Webseite verrücken, ausblenden, animieren.*

## Technisches Vorwissen

Um mit JavaScript beliebige Animationen oder DHTML-Anwendungen aufsetzen zu können, sollten Sie mit folgenden Techniken vertraut sein:

- Aufsetzen browserunabhängigen Codes (siehe Kapitel 10.5)

- Zugriff auf HTML-Elemente und deren Eigenschaften, einschließlich CSS-Stileigenschaften (siehe Kapitel 10. 4)
- freie Positionierung von HTML-Elementen (siehe Kapitel 5)
- Steuerung der Sichtbarkeit von HTML-Elementen (siehe Kapitel 5.5 und 10.7)
- Periodische Wiederholung von Code

Von all diesen Techniken fehlt uns nur noch die letzte.

## Periodische Wiederholung von Code

Die einfachste Möglichkeit, einen Code mehrfach hintereinander auszuführen, besteht in der Implementierung einer Schleife. Für die Implementierung von Animationen mit JavaScript sind Schleifen aber üblicherweise nicht sinnvoll, weil es keine vernünftige Möglichkeit gibt, zwischen den einzelnen Schleifeniterationen Pausen von festgelegter Dauer einzurichten.

Hingegen bietet uns JavaScript die Möglichkeit, eine Funktion nach Ablauf einer bestimmten Zeitspanne aufzurufen - und zwar mit Hilfe von `setTimeout()`.

```
setTimeout("eineFunktion()",200);
```

Die `window`-Methode `setTimeout()` übernimmt als Argumente den Namen einer JavaScript-Funktion und eine Zeitangabe in Millisekunden. Nach Ablauf dieser Zeitspanne wird die übergebene Funktion aufgerufen.

Nehmen wir nun an, wir wollten ein HTML-Element alle zwei Sekunden um zehn Pixel nach rechts bewegen. Um dies mit Hilfe von `setTimeout()` implementieren zu können, bedarf es eines kleinen Tricks: Man definiert eine Animations-Funktion, in der man das HTML-Element um zehn Pixel verrückt. Am Ende der Funktion ruft man `setTimeout()` mit 2000 (gleich 2 Sekunden) als zweitem und - jetzt kommt der Trick - der Animations- Funktion als erstem Argument auf.

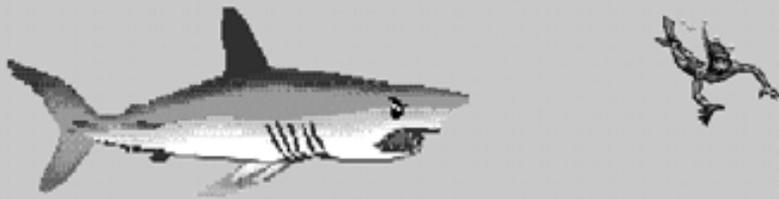
```
function animFunk()
{
 // Code der HTML-Element verrückt
 if (weitermachen)
 setTimeout("animFunk()",2000);
}
```

So sorgt die Funktion selbst dafür, dass sie in regelmäßigen Abständen aufgerufen (und das HTML-Elemente immer weiter gerückt) wird. Die `if`-Bedingung, die den Aufruf `setTimeout()` kontrolliert, wird benötigt, um die Animation irgendwann abubrechen.

## Die Animation

Die Animation, die wir in diesem Abschnitt erstellen werden, zeigt eine Studie zum Fressverhalten des Weißen Hais und sollte im Internet Explorer 4/5, Netscape Navigator 4 und Netscape 6-Browser laufen.

## Eine kleine Animation



## Eine kleine Animation



## Eine kleine Animation



**Abbildung 16.10: Drei Ausschnitte aus der Animation**

Schauen wir uns zuerst den reinen HTML-Code an:

### Listing 16.2: HaiAnimation.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>

<title>Der weiÙe Hai</title>

<style type="text/css">
 body { background-color: #cdcdcd }
 #hai { position: absolute; top: 100px; left: 20px; z-index: 2 }
 #opfer { position: absolute; top: 100px; left: 520px; z-index: 1 }
</style>
<script language="JavaScript">
// hier fehlt noch der Skriptcode
</script>
</head>
<body onload="animate()">
<h1>Eine kleine Animation</h1>
<div id="hai"></div>
```

```
<div id="opfer"></div>
</body>
</html>
```

In den letzten Zeilen fällt auf, dass die Bilder in <div>-Tags eingeschlossen wurden. Dies ist erforderlich, da der Netscape Navigator keine direkte Verschiebung von <img>- Elementen erlaubt.

Des Weiteren fällt auf, dass den <div>-Tags IDs zugewiesen wurden. Wir benötigen diese nicht nur zur Verknüpfung mit den Stylesheets aus dem Header-Bereich der Webseite, sondern auch später, um im JavaScript-Code auf den Hai und den Taucher zugreifen zu können.

Damit wären wir bei den Stylesheets. Wie Sie sehen, dienen diese dazu, die Bilder an ihre Anfangspositionen zu rücken. Die absolute Positionierung hat aber auch noch einen weiteren Effekt: Netscape Navigator speichert alle absolut positionierten Elemente in seinem layers[]-Array. Das layers[]-Array wurde zwar nie in den HTML-Standard aufgenommen und wird auch vom Netscape 6-Browser nicht mehr unterstützt, doch stellt es die einzige Möglichkeit dar, Animationen zu schreiben, die im Navigator laufen.

Beachten Sie auch die z-index-Werte, die dafür sorgen, dass der Hai über dem Taucher angezeigt wird, wenn sich beide Bilder überlagern. So können wir den Eindruck erwecken, dass der Hai den Taucher verschluckt.

Lassen Sie uns jetzt den Skript-Code durchgehen. Beim Laden der Webseite werden die Anweisungen im Skript-Tag ausgeführt.

```
<script language="JavaScript">
var isNav4 = false;
var isNet6 = false;
var isIE = false;
var loop = 1;
var haibilder = new Array();
 haibilder[0] = new Image();
 haibilder[0].src = "hai.gif";
 haibilder[1] = new Image();
 haibilder[1].src = "hai2.gif";
```

Wir haben Boolesche Variablen zur Unterscheidung der Browser definiert und auf false gesetzt. Die globale Variable loop soll die Anzahl der Iterationen zählen und die Animation nach 50 Iterationen abbrechen.

Schließlich laden wir die beiden Haidarstellungen (mit geschlossenem und offenem Maul in ein JavaScript-Array.

Wenn der Besucher der Website auf den Start-Schalter drückt, wird die JavaScript- Funktion animate() aufgerufen (siehe HTML-Code oben). Diese ruft ihrerseits als Erstes die Funktion init() auf, die feststellt, in welchem Browser die Webseite angezeigt wird und dementsprechend eine der oben definierten Booleschen Variablen auf true setzt.

```

function init()
{
var brw = navigator.appName.toLowerCase();
if (brw.indexOf("microsoft") != -1) // Internet Explorer
{
isIE = true;
}
else if (brw.indexOf("netscape") != -1) // Netscape
{
if (parseInt(navigator.appVersion) < 5) // Navigator 4
// oder früher
{
isNav4 = true;
}
else // Netscape 6
// oder später
{
isNet6 = true;
}
}
else
{
alert("Unbekannter Browser, Animation wird vermutlich nicht
funktionieren");
}
}

```

Damit wären wir bei der eigentlichen Animationsfunktion angelangt. Nach dem Aufruf von init() bewegt diese Hai und Taucher um einen Schritt weiter.

```

function animate()
{
init();
if(isIE)
{
document.all.hai.style.left = 20 * loop + "px";
document.all.opfer.style.top = 100 + 2*loop + "px";
document.all.opfer.style.left = 520 + 2*loop + "px";
}
else if (isNav4)
{
document.layers["hai"].left = 20 * loop;
document.layers["opfer"].top = 100 + 2*loop;
document.layers["opfer"].left = 520 + 2*loop;
}
else
{
var vhai = document.getElementById('hai');
vhai.style.left = 20 * loop + "px";
var vopfer = document.getElementById('opfer');

```

```
vopfer.style.top = 100 + 2 * loop + "px";
vopfer.style.left = 520 + 2 * loop + "px";
}
```

Die neuen Positionen werden als absolute Positionen berechnet. Der Einbau der Schleifenvariablen `loop` sorgt dafür, dass die Positionen von Iteration zu Iteration weiter vorgerückt werden. Für alle drei Browser erfolgt die Positionierung über die Eigenschaften `top` und `left`, die Browser unterscheiden sich allerdings darin, wie man auf diese Positionen zugreifen kann.

Im Internet Explorer nutzen wir die Tatsache, dass alle HTML-Elemente unter der proprietären `document`-Eigenschaft `all` verfügbar sind und über ihre ID - sofern sie über eine solche verfügen - angesprochen werden können. Die Eigenschaften `top` und `left` sind im Internet Explorer die CSS-Stylesheet-Eigenschaften und werden daher über das `style`-Unterobjekt angesprochen.

```
document.all.hai.style.left = 20 * loop + "px";
```

Im Netscape Navigator werden die absolut positionierten Elemente in das proprietäre `layers`-Array eingetragen und können mit ihrer ID als Index angesprochen werden. Der Navigator unterstützt keine CSS-Stileigenschaften, verfügt aber ebenfalls über Eigenschaften `top` und `left`, die man zur Positionierung verwenden kann (man hängt dann aber keine Einheit an den Zahlenwert an!).

```
document.layers["hai"].left = 20 * loop;
```

Der Netscape 6-Browser unterstützt wie der Internet Explorer die CSS-Stileigenschaften und das `style`-Unterobjekt. Allerdings muss man sich für den Zugriff auf die HTML-Elemente der DOM-Methoden, hier `getElementById()` bedienen.

```
var vhai = document.getElementById('hai');
vhai.style.left = 20 * loop + "px";
```

Jetzt sind beide Bilder vorgerückt. Als Vorbereitung auf die nächste Iteration inkrementieren wir den internen Zähler `loop`:

```
++loop;
```

Um die Animation ein wenig interessanter zu machen, tauschen wir im Internet Explorer und im Netscape 6-Browser nach 10 Iterationen das Hai-Bild (der Hai reißt dann das Maul auf).

```
if((loop > 10 && loop < 15) && (isIE || isNet6))
{
 document.images[0].src = haibilder[1].src;
}
```

Nach 15 Iterationen ist der Hai so weit vorgerückt, dass er den Taucher halb verdeckt. Damit der Taucher später nicht wieder hinter dem Hai auftaucht, verbergen wir nach 15 Iterationen das Taucher-Bild, indem wir die `visibility`-Eigenschaft auf `hidden` (verborgen) setzen. Wie zuvor bei

der Positionierung müssen wir dabei für die verschiedenen Browser unterschiedliche Wege gehen.

Im Internet Explorer und im Netscape 6-Browser schließen wir danach auch noch das Maul des Hais.

```
if(loop > 15)
{
 if(isIE)
 {
 document.all.opfer.style.visibility = "hidden";
 }
 else if (isNav4)
 {
 document.layers["opfer"].visibility = "hidden";
 }
 else
 {
 var vopfer = document.getElementById('opfer');
 vopfer.style.visibility = "hidden";
 }
 if (isIE || isNet6)
 document.images[0].src = haibilder[0].src;
}
```

Damit wären wir am Ende der Funktion. Jetzt müssen wir nur noch dafür sorgen, dass die animate()-Funktion so lange weiter aufgerufen wird, bis ca. 50 Iterationen durchlaufen sind. Dafür würde im Grunde eine if-Bedingung genügen. Wir wollen den Hai aber in der mittleren Bewegungsphase, wenn er das Maul aufreißt, schneller schwimmen lassen. Daher rufen wir setTimeout() in Abhängigkeit von der Iteration (Wert in loop) mit verschiedenen Zeitangaben auf.

```
if(loop < 8)
 setTimeout("animate()",200);
else if (loop < 18)
 setTimeout("animate()",100);
else if (loop < 50)
 setTimeout("animate()",200);
}
</script>
```

Damit hätten wir Ihnen nun gezeigt, wie man CSS-StyleSheets zur Animation nutzen kann. Die schwierigste und lästigste Arbeit ist dabei zweifelsohne die Unterstützung der verschiedenen Browser - und dabei kann man noch nicht einmal sicher sein, ob die Animation in den nächsten Browserversionen auch noch lauffähig ist. Es bleibt aber zu hoffen, dass sich die großen Browser mehr und mehr dem DOM-Standard annähern (Netscape hat dies in seinem Netscape 6-Browser ja schon getan) und wir uns die Fallunterscheidungen in baldiger Zukunft sparen können.

## JavaScript/DHTML versus GIF-Animationen

GIF-Animationen, sofern sie gut gemacht sind, laufen meist schneller und weniger ruckhaft als vergleichbare JavaScript-Animationen. Mit JavaScript kann man dafür

- beliebige HTML-Elemente animieren
- HTML-Elemente frei über die Webseite bewegen (während GIF-Animationen auf den Anzeigebereich ihres <img>-Tags begrenzt sind )
- Elemente aus- und einblenden (siehe auch Kapitel 10.7)
- auf Besucherereignisse reagieren

## 16.4 Java-Animationen

Ich hoffe, Sie haben es nicht versäumt, Kapitel 15 über die Erstellung von Java-Applets zu lesen. Denn ohne Grundwissen über den Aufbau von Applets, die Applet- Standardmethoden, die Auswertung von HTML-Parametern und die grundlegende Java- Syntax werden Sie in diesem Abschnitt nicht viel verstehen. Sie haben Kapitel 15 gelesen? Na prima, dann können wir uns gleich auf ein neues Konzept stürzen: die Threads.

### Threads

Das englische Wort »Thread« bedeutet soviel wie »Faden«. Wenn man diesen Faden als einen Handlungsfaden auffasst, bekommt man schon eine ganz gute Ahnung davon, worum es bei der Thread-Programmierung geht. Normalerweise besteht ein Applet (wie im Übrigen jedes beliebige Programm) aus einem einzigen Handlungsfaden, der beim Programmstart aufgenommen und Anweisung für Anweisung verfolgt und ausgeführt wird.

Manchmal wäre es aber von Vorteil, wenn ein Programm zwei Aufgaben gleichzeitig erledigen könnte. Schauen wir uns ein Beispiel aus der täglichen Praxis an. Sie sitzen an Ihrem Computer und bearbeiten in Ihrem Textverarbeitungsprogramm ein größeres Dokument. Das Textverarbeitungsprogramm verfügt über einen einzelnen Handlungsfaden, der ausgeführt wird, während Sie mit dem Programm arbeiten. Jetzt wollen Sie das Dokument auf dem aktuellen Bearbeitungsstand ausdrucken. Das Textverarbeitungsprogramm muss das Dokument (falls es sehr umfangreich ist und der Drucker nur über wenig eigenen Speicher verfügt) häppchenweise aufbereiten und an den Drucker schicken. Wenn Ihr Textverarbeitungsprogramm nicht threadfähig ist, konzentriert es sich jetzt ganz auf die Bearbeitung der Druckausgabe und Sie haben keine Chance, den Text, während er gedruckt wird, weiter zu bearbeiten - älteren Windows- Anwender dürfte ein solches Szenario wohl bekannt sein. ;-) Ist das Programm dagegen threadfähig, erzeugt es für das Ausdrucken einen neuen Thread. Dieser neue Thread oder Handlungsfaden bearbeitet die Druckerausgabe. Der Hauptthread des Programms wird davon nicht beeinträchtigt und kann weiter ausgeführt werden, das heißt, Sie können während des Druckens mit dem Programm normal weiter arbeiten.

#### Threads und das Betriebssystem

Multithreading ist nicht nur eine Frage der Implementierung eines Programms, sondern auch eine Frage des Betriebssystems. Dieses muss dafür ausgelegt sein, mehrere Programme oder Threads<sup>1</sup> gleichzeitig ausführen zu können. Den Begriff »gleichzeitig« darf man dabei nicht zu wörtlich nehmen, besser wäre es von quasi gleichzeitig oder quasiparallel zu reden.

Wenn ein Thread ausgeführt wird, belegt es den Prozessor des Rechners. Hier, im Prozessor, wird der Code des Threads abgearbeitet und ausgeführt. Hat ein Rechner nur einen einzigen Prozessor, kann aber auch immer nur ein Thread gleichzeitig ausgeführt werden. Wie ist dann aber die gleichzeitige Ausführung von mehreren Threads möglich?

Wann immer ein Thread gestartet wird, nimmt das Betriebssystem den Thread in Empfang und reiht ihn in eine schön geordnete Warteschleife ein. Das Betriebssystem geht die Warteschleife in einem endlosen Zyklus durch, greift sich einen nach dem anderen die Threads heraus und stellt ihnen für eine bestimmte Zeit den Prozessor zur Verfügung. Während ein Thread über den Prozessor verfügt, wird sein Code ausgeführt. Nach kurzer Zeit klopft dann aber das Betriebssystem an, gibt Bescheid, dass die Zeit abgelaufen ist, und teilt dem nächsten Thread den Prozessor zu, während sich der alte Thread wieder in die Warteschleife einreihet. So werden die einzelnen Threads nacheinander immer Stück für Stück ausgeführt. Ist der Rechner schnell genug, läuft diese häppchenweise Ausführung der Threads aber so schnell ab, dass der Anwender gar nichts davon merkt, sondern das Gefühl hat, dass die einzelnen Threads (oder Programme) parallel ausgeführt werden.

Diese quasiparallele Ausführung ist eine Eigenschaft des Betriebssystems. Multithreadfähige Betriebssysteme sind Unix/Linux sowie Windows 95/98/NT/2000, aber nicht Windows 3.x. Die Aufspaltung eines Programms in mehrere Threads ist dagegen eine Frage der Implementierung des Programms.

1

Soweit es die parallele Ausführung betrifft, gibt es im Grunde keinen Unterschied zwischen Programmen und Threads. Ein Programm ist in diesem Sinne einfach ein Hauptthread mit keinem, einem oder mehreren weiteren Threads.

## Threads für Java-Applets

Wenn wir Animationen in Java-Applets implementieren, nutzen wir die Möglichkeiten der Thread-Programmierung. Zum einen können die Animationen dann parallel zum Hauptthread des Applets ausgeführt werden, zum anderen können wir mit Hilfe der für Threads verfügbaren Methoden die Animationen gezielt starten, zeitweise anhalten oder ganz stoppen.

## Thread-Programmierung in Java

In Java gibt es zwei Möglichkeiten, Threads zu implementieren:

- durch Ableitung von der Klasse Thread oder
- durch Implementierung der Schnittstelle Runnable

## Die Klasse Thread

Die Realisierung eines Threads ist zunächst ganz einfach. Man leitet eine eigene Klasse von Thread ab

```
class MeineKlasse extends Thread
{
```

und erbt dadurch eine Reihe von interessanten Methode, die zur Steuerung eines Threads benötigt werden (siehe Tabelle 16.1).

Methoden	Beschreibung
<code>run()</code>	Diese Methode ist das Herz des Threads. Hierhin gehören alle Anweisungen, die er während seines Lebens ausführen soll. Die <code>run()</code> -Methode bestimmt auch die Lebensdauer des Threads: wenn sie beendet wird, endet auch der Thread. Die <code>run()</code> -Methode dürfen Sie nicht direkt aufrufen, da dann kein neuer Thread im Betriebssystem angelegt wird.
<code>start()</code>	Diese Methode rufen Sie als Programmierer auf, um einen Thread zu starten. Es werden einige interne Initialisierungen durchgeführt und dann <code>run()</code> aktiviert.
<code>boolean isAlive()</code>	Liefert <code>true</code> , wenn der Thread gestartet worden ist und noch am Laufen ist.
<code>Thread currentThread()</code>	Liefert eine Referenz auf den Thread, der gerade vom Prozessor abgearbeitet wird. Die Methode kann ohne Instanz als <code>Thread.currentThread()</code> aufgerufen werden.
<code>sleep(long z)</code>	Veranlasst, dass sich der Thread für <code>z</code> Millisekunden schlafen legt. Der Aufruf muss in einem try-catch Block geklammert werden. Die Methode kann ohne Instanz aufgerufen werden. In diesem Fall legt sich der den Aufruf umgebende Thread schlafen.
<code>yield()</code>	Der aufrufende Thread gibt freiwillig den Prozessor ab und legt sich schlafen, bis er wieder vom Laufzeitsystem den Prozessor erhält.

**Tabelle 16.1: Thread-Methoden**

## Die Schnittstelle Runnable

Will man Applets threadfähig machen, stößt man auf das Problem, dass man die Appletklasse nicht von der Klasse Thread ableiten kann, weil sie ja bereits von der Basisklasse Applet abgeleitet wird (und Java keine Ableitung von zwei Basisklassen erlaubt). Dass man Applets dennoch threadfähig machen kann, verdanken wir der Schnittstelle Runnable.



*Eine Schnittstelle kann man sich vielleicht am einfachsten als eine Klasse vorstellen, in der eine oder mehrere Methoden deklariert, aber nicht definiert sind. Das heißt, die Schnittstelle gibt an, wie die Methoden heißen und welche Parameter sie übernehmen, aber sie definiert keine Anweisungsblöcke für die Methoden. Schnittstellen sind für Java-Klassen so etwas wie Zertifikate. Wenn Sie sich irgendwo als Webdesigner oder Programmierer bewerben wollen, müssen Sie nachweisen, dass Sie über Erfahrungen in Webdesign, JavaScript oder einer Programmiersprache wie Java verfügen. Dazu belegen Sie einen Kursus oder ein Studium, das Sie mit einem Zertifikat abschließen. Wenn eine Java-Klasse am Thread-Mechanismus teilhaben will, leitet sie sich von der Klasse Thread ab oder implementiert die Schnittstelle Runnable und weist sich dadurch als threadfähig aus.*

Eine Klasse kann sich von beliebig vielen Schnittstellen ableiten. Man spricht dann allerdings nicht von Ableitung, sondern von Implementierung, denn eine Klasse, die eine Schnittstelle benutzt, muss alle in der Schnittstelle deklarierten Methoden implementieren (das heißt, sie muss Anweisungsblöcke für die Methoden definieren). Im Falle der Schnittstelle Runnable ist dies nur eine Methode: run().

## Threadfähige Applets

Was also ist zu tun, wenn man ein Applet threadfähig machen will?

1. Zuerst geben Sie in der Klassendefinition an, dass die Applet-Klasse die Schnittstelle Runnable implementiert.

```
public class MeinApplet extends Applet implements Runnable
{
```

2. Dann definieren Sie in der Klasse die Methode run(). In den Anweisungsblock dieser Methode schreiben Sie alle Anweisungen, die ausgeführt werden sollen, wenn der Thread gestartet wird.

```
 public void run()
 {
 // Anweisungen
 }
```

3. Bis jetzt haben Sie aber noch keinen neuen Thread. Diesen erzeugen Sie, indem Sie mit Hilfe des new-Operators ein Thread-Objekt anlegen. Dem Konstruktor der Klasse Klasse Thread übergeben Sie dabei das Schlüsselwort this, das im Code der Applet- Klasse für das Applet selbst steht. Der Java-Compiler weiß dann, dass er die run()- Methode des Thread-Objekts nicht in dem Objekt selbst, sondern in der Applet-Klasse findet.

```
public class Laufschrift extends Applet implements Runnable
{
 Thread m_thread = null;
 ...
}
```

```

public void start()
{
 if (m_thread == null)
 {
 m_thread = new Thread(this);
 ...
 }
}

```

4. Zu guter Letzt starten Sie den Thread, indem Sie die start()-Methode des Thread- Objekts aufrufen.

```

public void start()
{
 if (m_thread == null)
 {
 m_thread = new Thread(this);
 m_thread.start();
 }
}

```

Wie dies in der Praxis aussieht, demonstriert der folgende Abschnitt.

## Laufschriften (Ticker)

Das folgende Applet erzeugt eine Laufschrift, die endlos von rechts nach links über das Anzeigefeld des Applets läuft. Über spezielle Parameter kann man vom HTML-Code aus den Text, die Farben und die Schriftgröße der Laufschrift einstellen.

### Listing 16.3: Laufschrift.java

```

// Laufschrift-Applet
import java.awt.*;
import java.applet.*;
public class Laufschrift extends Applet implements Runnable
{
 Thread m_thread = null;

 String m_text;
 int m_breite;
 int m_hoehe;
 int m_hg_rot, m_hg_gruen, m_hg_blau;
 int m_vg_rot, m_vg_gruen, m_vg_blau;
 int m_font_groesse;
 int x, y;

 public void init()
 {
 m_text = getParameter("text");
 m_breite = Integer.valueOf(getParameter("breite")).intValue();
 m_hoehe = Integer.valueOf(getParameter("hoehe")).intValue();
 }
}

```

```

m_hg_rot = Integer.valueOf(getParameter("hg_rot")).intValue();
m_hg_gruen = Integer.valueOf(getParameter("hg_gruen")).intValue();
m_hg_blau = Integer.valueOf(getParameter("hg_blau")).intValue();
m_vg_rot = Integer.valueOf(getParameter("vg_rot")).intValue();
m_vg_gruen = Integer.valueOf(getParameter("vg_gruen")).intValue();
m_vg_blau = Integer.valueOf(getParameter("vg_blau")).intValue();
m_font_groesse =
 Integer.valueOf(getParameter("font_size")).intValue();

x = m_breite;
y = m_hoehe/2;
setBackground(new Color(m_hg_rot, m_hg_gruen, m_hg_blau));
setForeground(new Color(m_vg_rot, m_vg_gruen, m_vg_blau));
setFont(new Font("Monospaced", Font.BOLD, m_font_groesse));
}
public void start()
{
 if (m_thread == null)
 {
 m_thread = new Thread(this);
 m_thread.start();
 }
}
public void stop()
{
 if (m_thread != null)
 {
 m_thread.interrupt();
 m_thread = null;
 }
}
public void run()
{
 while (m_thread == Thread.currentThread())
 {
 try
 {
 repaint();
 Thread.sleep(100);
 }
 catch (InterruptedException e)
 {
 return;
 }
 }
}
public void paint(Graphics gc)
{
 FontMetrics fm = gc.getFontMetrics();

```

```

gc.drawString(m_text, x, y);
x -= 5;
if(x < -fm.stringWidth(m_text))
 x = m_breite;
}
}

```

Lassen Sie uns gemeinsam den Lebenslauf dieses Applets verfolgen.

Wenn eine Webseite mit diesem Applet in einen Browser geladen wird, richtet das Applet zuerst einmal Speicher für eine Reihe von Instanzvariablen ein: eine Instanzvariable `m_thread` für den Thread, eine Reihe von Instanzvariablen für die Argumente, die vom HTML-Code übergeben werden und zwei Instanzvariablen `x` und `y`, die angeben, wo die Laufschrift in jedem Schritt einzuzeichnen ist.

```

Thread m_thread = null;

String m_text;
int m_breite;
int m_hoehe;
int m_hg_rot, m_hg_gruen, m_hg_blau;
int m_vg_rot, m_vg_gruen, m_vg_blau;
int m_font_groesse;
int x, y;

```

Dann ruft der Browser die Applet-Methode `init()` auf.

```

public void init()
{
 m_text = getParameter("text");
 m_breite = Integer.valueOf(getParameter("breite")).intValue();
 m_hoehe = Integer.valueOf(getParameter("hoehe")).intValue();
 m_hg_rot = Integer.valueOf(getParameter("hg_rot")).intValue();
 m_hg_gruen = Integer.valueOf(getParameter("hg_gruen")).intValue();
 m_hg_blau = Integer.valueOf(getParameter("hg_blau")).intValue();
 m_vg_rot = Integer.valueOf(getParameter("vg_rot")).intValue();
 m_vg_gruen = Integer.valueOf(getParameter("vg_gruen")).intValue();
 m_vg_blau = Integer.valueOf(getParameter("vg_blau")).intValue();
 m_font_groesse =
 Integer.valueOf(getParameter("font_size")).intValue();

 x = m_breite;
 y = m_hoehe/2;
 setBackground(new Color(m_hg_rot, m_hg_gruen, m_hg_blau));
 setForeground(new Color(m_vg_rot, m_vg_gruen, m_vg_blau));
 setFont(new Font("Monospaced", Font.BOLD, m_font_groesse));
}

```

In dieser Methode liest das Applet die HTML-Argumente ein. Ein passender HTML-Code zur

Einbindung des Applets würde beispielsweise wie folgt aussehen:

#### Listing 16.4: Auszug aus Laufschrift.html

```
<applet code="Laufschrift.class" width="600" height="40">
 <param name="text" value="Heute Zucchini im Sonderangebot" />
 <param name="breite" value="600" />
 <param name="hoehe" value="40" />
 <param name="hg_rot" value="255" />
 <param name="hg_gruen" value="0" />
 <param name="hg_blau" value="0" />
 <param name="vg_rot" value="255" />
 <param name="vg_gruen" value="255" />
 <param name="vg_blau" value="255" />
 <param name="font_size" value="18" />
</applet>
```

Der Text für die Laufschrift wird in der Instanzvariablen `m_text` gespeichert.

Die aktuellen Anzeigepositionen für die Laufschrift, `x` und `y`, werden so gesetzt, dass die Laufschrift am rechten Rand in mittlerer Höhe des Laufbandes (der Anzeigefläche des Applets) beginnt.

Das Setzen der Hintergrund- und Vordergrundfarbe ist etwas umständlich, da es kein einfaches Verfahren zur Umwandlung von Strings mit hexadezimalen Werten (beispielsweise »0xFF0000«, wie Sie `getParameter()` zurückliefern würde) in hexadezimale Integer-Werte (wie sie der Konstruktor der Klasse `Color` erwartet) gibt. Aus diesem Grunde übergeben wir für jeden Farbanteil der RGB-Farben einen eigenen Integer-Wert im Bereich zwischen 0 und 255.

Schließlich wird noch eine neue proportionale, fette Schrift in der angegebenen Größe erzeugt und als Standardschrift für die Applet-Ausgabe eingerichtet (Aufruf von `setFont()`).

Nach der `init()`-Methode ruft der Browser die `start()`-Methode auf.

```
public void start()
{
 if (m_thread == null)
 {
 m_thread = new Thread(this);
 m_thread.start();
 }
}
```

In der `start()`-Methode prüfen wir, ob bereits ein `Thread`-Objekt existiert. Wenn dies nicht der Fall ist, liefert die `if`-Bedingung `true` zurück und wir erzeugen ein neues `Thread`-Objekt und starten den `Thread` durch Aufruf seiner `start()`-Methode. Wie Sie aus der Einleitung zu diesem Abschnitt wissen, wird daraufhin automatisch die `run()`-Methode des `Thread`s aufgerufen. Da wir dem `Thread`-Konstruktor das Schlüsselwort `this` übergeben haben, das unser Applet repräsentiert, ist

dies die run()-Methode aus unserem Applet.

```
public void run()
{
 while (m_thread == Thread.currentThread())
 {
 try
 {
 repaint();
 Thread.sleep(100);
 }
 catch (InterruptedException e)
 {
 return;
 }
 }
}
```

Die run()-Methode besteht im Wesentlichen aus einer while-Schleife, die so lange ausgeführt wird, wie das Applet angezeigt wird. In der Schleife wird das nächste Bild der Animation gezeichnet (Aufruf von repaint()). Danach wird der Thread für 100 Millisekunden schlafen gelegt (Aufruf von Thread.sleep(100);) - ansonsten würde die Schrift so schnell animiert, dass man sie nicht lesen könnte.

Kompliziert wird die Sache durch zwei Umstände:

- Irgendwann muss die Schleife beendet werden. Dazu bedient man sich eines Tricks: man prüft, ob die Instanzvariable m\_thread noch auf den aktuellen Thread weist (zurückgeliefert von Thread.currentThread()). Ist dies der Fall, wird die run()-Methode weiter ausgeführt, ansonsten wird sie beendet (siehe hierzu auch die Implementierung der stop()-Methode.)
- Die sleep()-Anweisung muss in eine try-catch-Verzweigung eingebettet werden. Dies hat mit der Exception-Behandlung unter Java zu tun, auf die wir hier allerdings nicht näher eingehen wollen. Wir nehmen die try-catch-Syntax einfach so hin, wie sie ist.

Die run()-Methode ruft alle Zehntelsekunde die Methode repaint() auf, die ihrerseits die paint()-Methode des Applets aufruft.

```
public void paint(Graphics gc)
{
 FontMetrics fm = gc.getFontMetrics();
 gc.drawString(m_text, x, y);
 x -= 5;
 if(x < -fm.stringWidth(m_text))
 x = m_breite;
}
```

Die paint()-Methode lässt sich zuerst ein FontMetrics-Objekt zurückliefern, über das Sie Informationen zu der verwendeten Schrift abfragen kann. Dann zeichnet Sie den Schriftzug

(Inhalt von `m_text`) an den Koordinaten `x`, `y`.

Damit die Schrift beim nächsten Mal weiter rechts eingezeichnet wird, wird der Wert von `x` anschließend um 5 verringert.

Jetzt müssen wir nur noch dafür sorgen, dass die Animation wieder von vorne beginnt, wenn die Schrift am rechten Rand entschwinden ist. Dazu lassen wir uns von der `FontMetrics`-Methode `stringWidth()` die Breite des Textes in `m_text` zurückliefern. Wenn `x` kleiner ist als die negative Breite des Textes, ist der Text ganz am linken Rand verschwunden. Dann setzen wir `x` zurück auf die Breite des Applet-Anzeigefeldes.

Irgendwann verlässt der Besucher die Webseite. Dann ruft der Browser die `stop()`-Methode des Applets auf.

```
public void stop()
{
 if (m_thread != null)
 {
 m_thread.interrupt();
 m_thread = null;
 }
}
```

In der `stop()`-Methode prüfen wir, ob der Thread noch ausgeführt wird. Wenn ja (if-Bedingung liefert `true`), wird der Thread abgebrochen (Aufruf von `interrupt()`) und unsere Thread-Variable `m_thread` setzen wir auf `null`. So erreichen wir, dass die `run()`-Methode sofort beendet wird, und die `start()`-Methode kann anhand des Wertes von `m_thread` feststellen, ob sie einen neuen Thread erstellen soll oder nicht (siehe oben).



Abbildung 16.11: Java-Laufschrift

## 16.5 Flash-Animationen

Macromedia Flash ist eines der populärsten Grafikprogramme zur Erstellung interaktiver Grafiken und Animationen für das Web (obwohl es nicht unter Unix/Linux läuft). Es wäre hoffnungslos, wollten wir im Rahmen dieses Buches versuchen, Ihnen Flash mit all seinen Möglichkeiten und Konzepten in nur halbwegs angemessener Ausführlichkeit vorzustellen. Wir werden uns daher auf das Thema beschränken, das uns im Rahmen dieses Abschnitts am meisten interessiert: die Erstellung von Animationen mit Flash. Wir werden zwar auch dieses Thema nicht vollständig behandeln können, aber wir können Sie in die grundlegenden Techniken einführen, mit Ihnen zusammen erste Animationen erstellen, schauen wie man mit Hilfe von Flash die Erstellung von Animationen automatisieren kann und zum Abschluss eine schon recht komplexe Animation implementieren. Danach sollten Sie in der Lage sein, mit Flash eigene einfache Animationen aufzusetzen oder sich mit Hilfe weiterführender Literatur tiefer in Flash einzuarbeiten.



*Eine aktuelle Trial-Version kann man sich von <http://www.macromedia.com> herunterladen.*

# Flash-Animationen erstellen

Als erste Animation wollen wir ein Standardbeispiel für Animationstechnik nachstellen: den hüpfenden Ball.

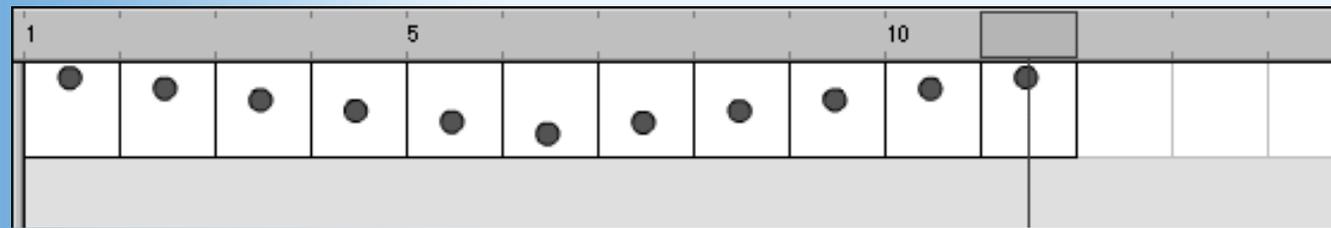


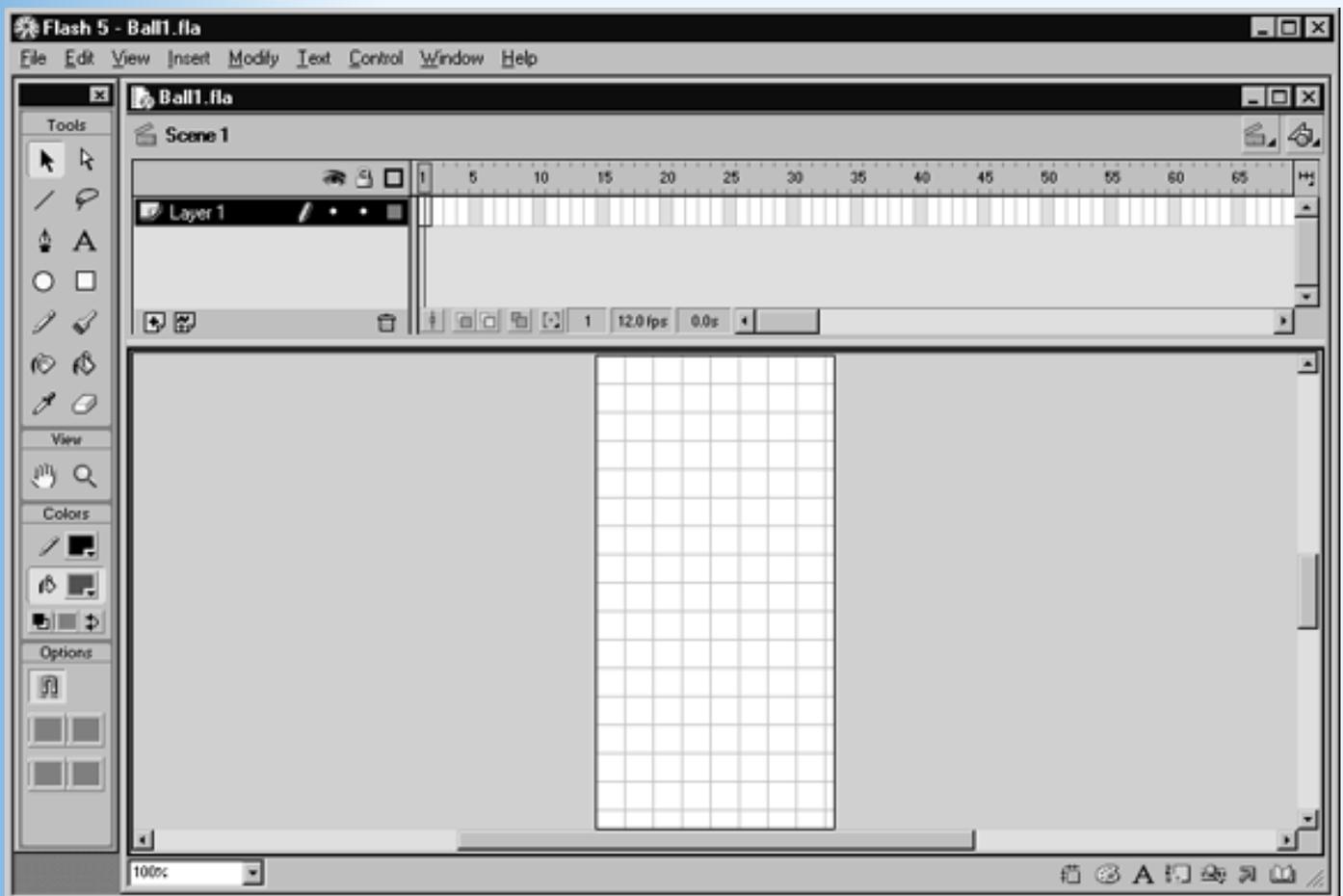
Abbildung 16.12: Hüpfender Ball

## Beginn einer neuen Animation

1. Starten Sie jetzt Flash. Flash legt automatisch ein neues Arbeitsdokument namens **Movie1** an.

Wenn Sie selbst ein neues Dokument (sprich eine neue Animation) beginnen wollen, rufen Sie den Befehl **File/New** auf.

2. Schließen Sie alle Fenster außer dem **Tools**-Fenster und dem Fenster mit dem Titel Movie1 (in der deutschen Version **Film1**).
3. Rufen Sie den Befehl **Modify/Movie** auf. Setzen Sie im Dialogfenster (siehe auch Abbildung 16.16) die Abmaße der Leinwand auf 150 px Breite (**Width**) mal 300 px Höhe (**Height**). Die Abmaße der Filmleinwand sind gleichzeitig die Abmaße der einzelnen Frames. Für unseren hüpfenden Ball ist eine schmale, hohe Leinwand bestens geeignet.
4. Aktivieren Sie die Menüoption **View/Grid/Show Grid**. In der Leinwand erscheint ein Hilfsgitter, das Ihnen das Zeichnen erleichtern kann.
5. Speichern Sie die Animation mit Hilfe des Befehls **File/Save** unter dem Namen *Ball1 fla*.



**Abbildung 16.13: Einrichtung der Arbeitsfläche**

Das Dokumentfenster **Ball1.fla** ist in zwei Bereiche geteilt:

- Der Zeitleiste (**Timeline**), in der Sie - nebeneinander aufgereiht - leere Schablonen für die nacheinander abzuspielenden Frames Ihrer Animation sehen
- Dem eigentlichen Arbeitsbereich (**Work Area**), in dem Sie den aktuellen Frame sehen (und dessen Anzeige über das Zoom-Feld links unten vergrößert oder verkleinert werden kann).



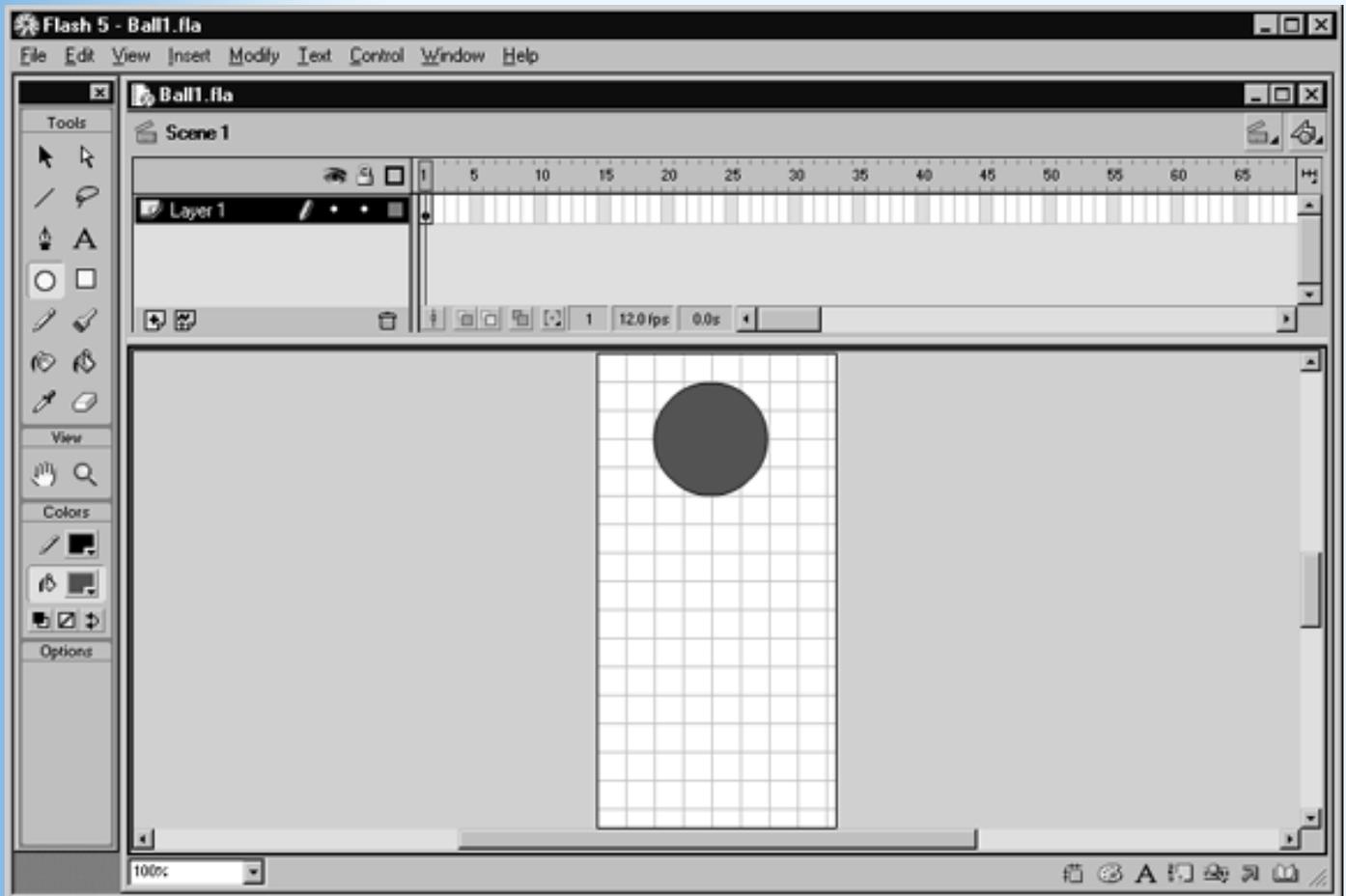
*Beide Bereiche kann man über die entsprechenden Befehle im Menü **View** ein- und ausblenden.*

## Frames zeichnen

1. Markieren Sie in der Zeitleiste den ersten Frame.
2. Wählen Sie im **Tools**-Fenster das Oval-Werkzeug aus. Klicken Sie danach mit der Maus in die Arbeitsfläche des Frames und ziehen Sie einen Kreis auf.
3. Wählen Sie im **Tools**-Fenster das Oval-Werkzeug aus. Klicken Sie danach mit der Maus in die Arbeitsfläche des Frames und ziehen Sie einen Kreis auf (siehe Abbildung 16.14).

Umriss- und Füllfarbe können Sie in der Tools-Leiste im Bereich **Colors** festlegen.

In der Zeitleiste wird jetzt im ersten Frame ein Kreis angezeigt. Dies ist nicht unser Ball, sondern ein Symbol, das anzeigt, dass der Frame ein nicht mehr leerer Schlüsselframe ist (mehr dazu weiter unten).



**Abbildung 16.14: Ball an der Startposition**

4. Kopieren Sie den Ball in den nächsten Frame.

Klicken Sie dazu mit der rechten Maustaste in den nächsten Frame in der Zeitleiste und wählen Sie den Befehl **Insert Keyframe** aus.

5. Jetzt verschieben wir den Ball in dem zweiten Frame ein Stück nach unten (beispielsweise um zwei Gitterkästchen). Wählen Sie in der Tools-Leiste das schwarze Zeiger-Symbol. Klicken Sie danach in den Ball im Frame und verschieben Sie ihn mit gedrückt gehaltener Maus nach unten.
6. Legen Sie auf diese Weise circa 11 Schlüsselframes (Keyframes) an, in denen der Ball zuerst nach unten und dann wieder nach oben verschoben ist.

Um besser kontrollieren zu können, wie sich die Animation von Frame zu Frame verändert, bietet Flash Ihnen die Möglichkeit, die benachbarten Frames in aufgehellten Farben mit einzublenden. Sie müssen dazu nur unten in der Zeitleiste den Schalter **Onion Skin** (Zwiebelhaut) drücken und in der über den Frames gelegenen Skala einstellen, wie viele Frames die Zwiebelhautdarstellung umfassen soll (siehe Abbildung 16.15).

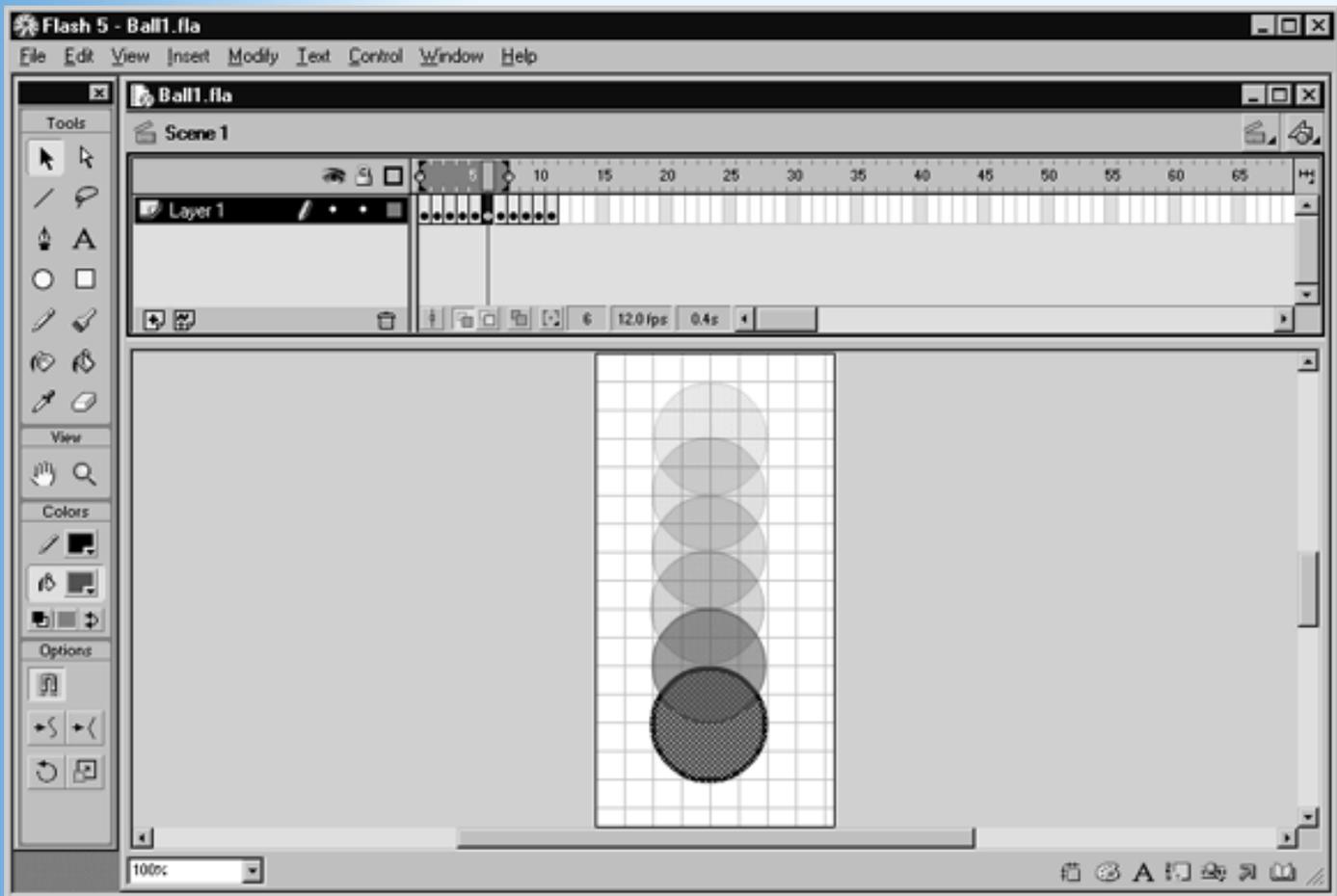


Abbildung 16.15: Zwiebildarstellung der Animation

7. Speichern Sie die Animation (**File/Save**).

## Animation testen

Zum Testen der Animation gibt es verschiedene Möglichkeiten:

- Rufen Sie den Befehl **Control/Play** auf (oder drücken Sie einfach nur die (Return)-Taste).
- Klicken Sie auf das Skalen-Symbol in der rechten oberen Ecke der Zeitleiste und wählen Sie den Befehl **Preview in Context** (statt **Normal**) aus.
- Rufen Sie den Befehl **Control/Test Movie** auf (oder drücken Sie einfach (Strg)+(Return)).

Vielleicht läuft Ihnen die Animation zu schnell ab. Dann rufen Sie noch einmal den Befehl **Modify Movie** auf und ändern Sie die Frame-Rate (Anzahl der Bilder pro Sekunde).

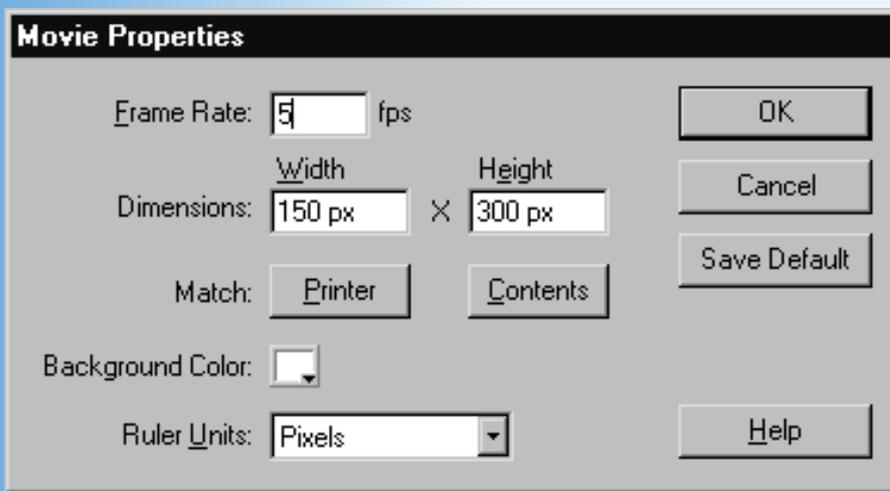


Abbildung 16.16: Animationseigenschaften setzen

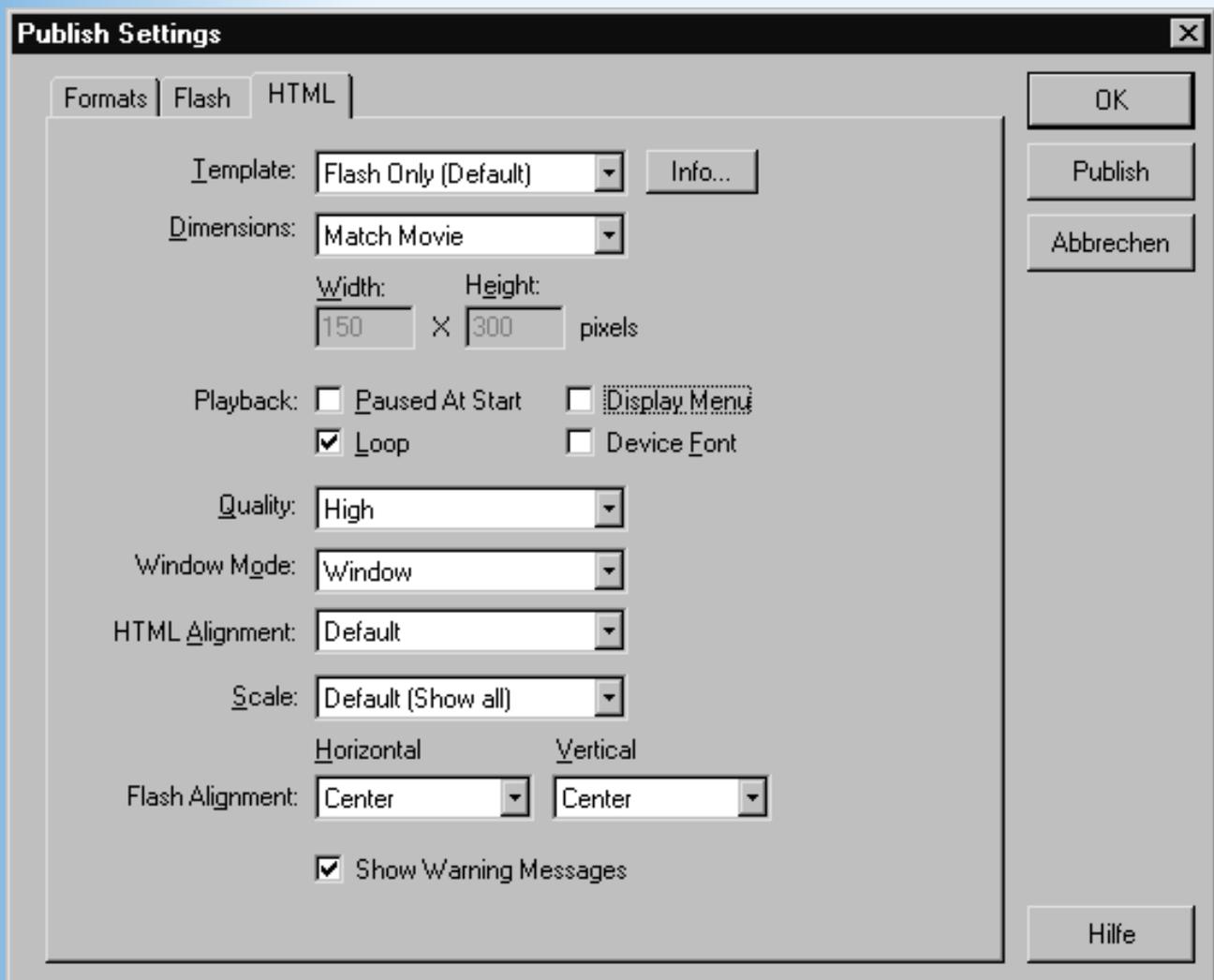
## Flash-Animationen in Webseiten einbinden

Flash-Animationen werden mit Hilfe des <object>- beziehungsweise des <embed>-Tags in Webseiten eingebettet. Sie brauchen den betreffenden Code nicht selbst aufzusetzen, lassen Sie ihn lieber direkt von Flash generieren.

1. Speichern Sie Ihre Animation.
2. Rufen Sie den Befehl **File/Publish Settings** auf.

Achten Sie darauf, dass auf der Registerkarte **Formats** die Option **HTML** gesetzt ist. Wechseln Sie dann zur Registerkarte **HTML** und legen Sie dort fest, wie die Flash- Animation in die Webseite eingebettet werden soll - üblicherweise brauchen Sie nur die Optionen unter **Playback** zu bearbeiten (siehe Abbildung 16.17).

3. Klicken Sie dann auf den Schalter **Publish**.



**Abbildung 16.17: Parameter für die Einbettung in HTML-Code**

Danach finden Sie in dem Verzeichnis, in dem Sie die Flash-Animation (.fla) abgespeichert haben,

- eine swf-Datei (die Flash-Animation im binären Export-Format) und
- eine .html-Datei, die die Animation so einbindet, dass sie sowohl im Internet Explorer als auch den Netscape-Browsern abgespielt werden kann.



*Voraussetzung ist allerdings, dass die Browser das Flash-Abspielplugin (den frei verfügbaren Flash-Player) installiert haben (das mittlerweile aber standardmäßig in den großen Browsern integriert ist).*

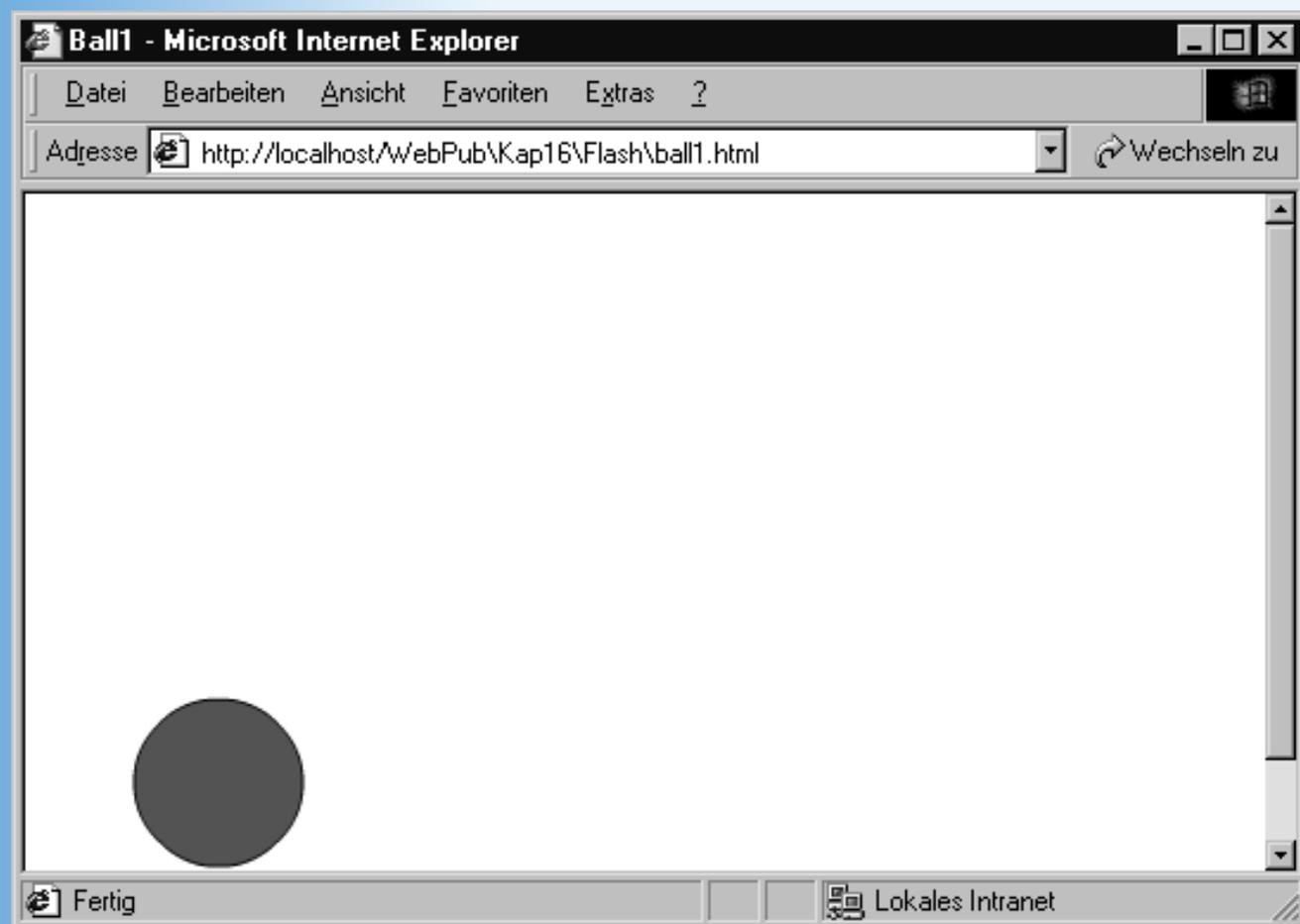
**Listing 16.5: Von Flash erzeugte HTML-Datei: ball1.html**

```
<HTML>
<HEAD>
<TITLE>Ball1</TITLE>
```

```

</HEAD>
<BODY bgcolor="#FFFFFF">
<!-- URL's used in the movie-->
<!-- text used in the movie-->
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
 codebase="http://download.macromedia.com/pub/shockwave/cabs
 -/flash/swflash.cab#version=5,0,0,0"
 WIDTH=150 HEIGHT=300>
 <PARAM NAME=movie VALUE="Ball1.swf">
 <PARAM NAME=menu VALUE=false>
 <PARAM NAME=quality VALUE=high>
 <PARAM NAME=bgcolor VALUE=#FFFFFF>
 <EMBED src="Ball1.swf" menu=false quality=high bgcolor=#FFFFFF
 WIDTH=150 HEIGHT=300 TYPE="application/x-shockwave-flash"
 PLUGINSOURCE="http://www.macromedia.com/shockwave/download/
 -index.cgi?P1_Prod_Version=ShockwaveFlash">
 </EMBED>
</OBJECT>
</BODY>
</HTML>

```



**Abbildung 16.18: Flash-Animation in Browser**

Das <object>-Tag enthält den Code für die Einbettung in den Internet Explorer (wobei der Flash-Player über eine eindeutige Objekt-ID spezifiziert wird). Das <embed>-Tag bindet die Animation so ein, dass sie vom Netscape Navigator und vom Netscape 6-Browser identifiziert und

abgespielt werden kann. Durch die Einbettung des <embed>-Tags in das <object>-Tag wird sichergestellt, dass alle drei Browser den für sie relevanten Code finden und nur diesen ausführen.

Sie können die von Flash erzeugte Webseite weiter bearbeiten oder den Code für die Einbettung der Animation in eine andere Webseite kopieren.

## Automatische Erzeugung von Zwischenbildern

Man kann sich die Erstellung von Animationen in Flash wesentlich erleichtern, indem man nur die Schlüsselframes einer Animation vorgibt und die dazwischenliegenden Frames automatisch von Flash generieren lässt. Was aber sind in diesem Sinne Schlüsselframes?

Ein Schlüsselframe ist in diesem Sinne jeder Frame, in dem die Animation (sprich die Bewegung des animierten Objekts) eine andere Richtung nimmt. Im Falle unseres hüpfenden Balls sind die Schlüsselframes schnell identifiziert: es sind

- der erste Frame, in dem die Bewegung nach unten beginnt
- der Frame, in dem der Ball ganz unten ankommt und nach dem die Aufwärtsbewegung einsetzt
- der letzte Frame, in dem die Bewegung nach oben endet

Im Folgenden werden Sie sehen, wie man die »Hüpfender Ball«-Animation durch Definition der drei Schlüsselframes und einer Technik namens Motion-Tweening erzeugt.

## Neue Animation beginnen

1. Schließen Sie die alte Animation, indem Sie den Befehl **File/Close** aufrufen.
2. Beginnen Sie eine neue Animation mit **File/New**.
3. Richten Sie die Animation wie im Abschnitt 16.5.1 ein.
4. Speichern Sie die neue Animation unter dem Namen *Ball2 fla*.

## Bewegungssequenz erzeugen

5. Zeichnen Sie wie in Abschnitt 16.5.1 einen Ball in den ersten Frame.

Der erste Frame wird - zusammen mit dem neuen Dokument - von Flash angelegt und ist automatisch ein Schlüsselframe.

6. Um den Ball von Flash animieren zu lassen, müssen wir ihn zu einem Symbol machen.

Markieren Sie dazu den gezeichneten Ball im Arbeitsbereich und rufen Sie den Menübefehl **Insert/Create Motion Tween** auf.

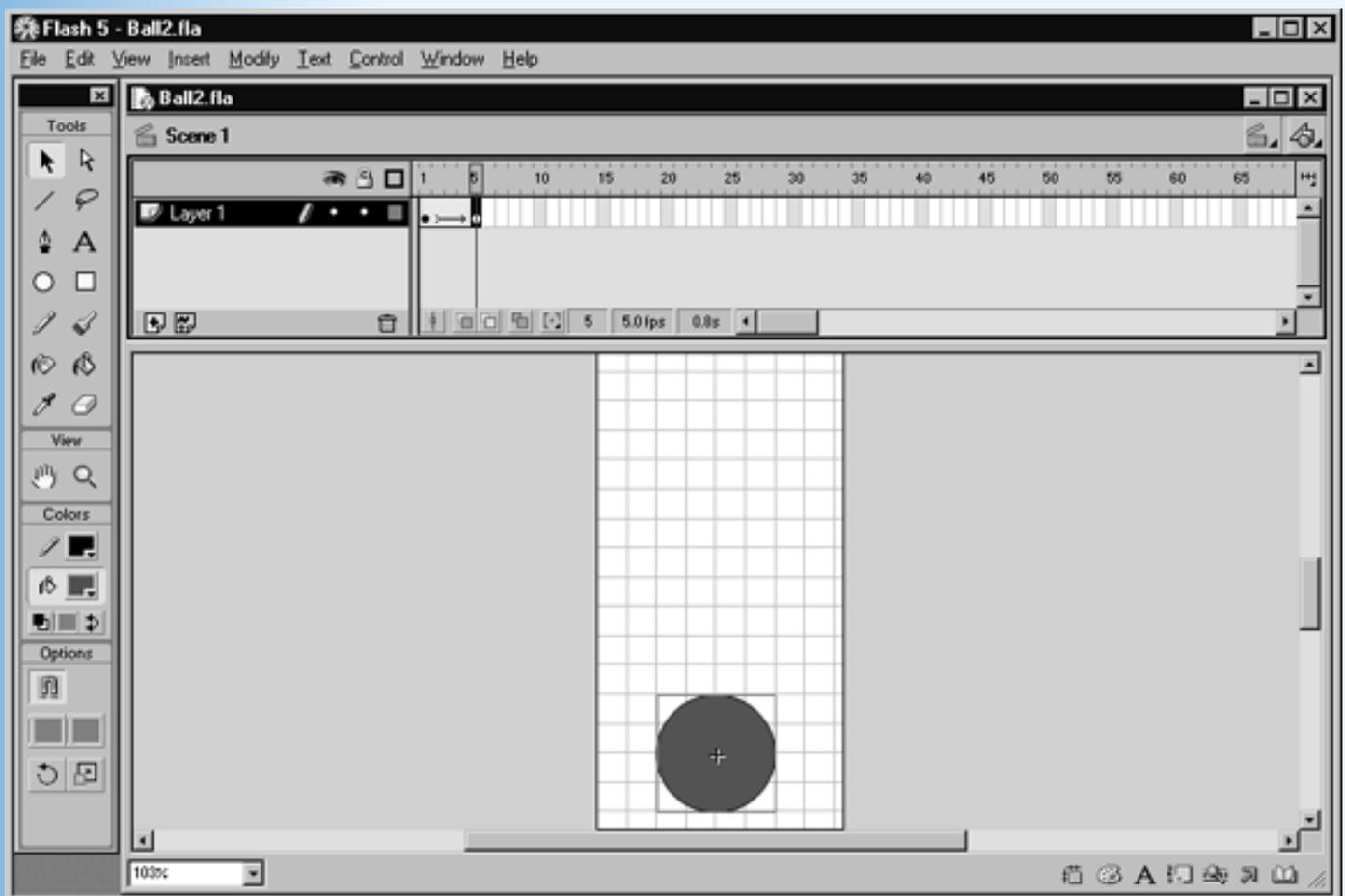


Die für eine Animation eingerichteten Symbole können Sie sich über den Befehl **Window/Library** anzeigen lassen.

7. Jetzt müssen wir den Endpunkt des Motion Tweens als neuen Schlüsselframe festlegen. Klicken Sie in der Zeitleiste mit der rechten Maustaste in den - sagen wir - fünften Frame und wählen Sie den Befehl **Insert Keyframe** aus.

Ziehen Sie dann den Ball im fünften Frame nach unten, und Flash erzeugt in den zwischen den Schlüsselframes gelegenen Frames die Zwischenbilder der Animation.

In der Zeitleiste wird der Motion Tween-Bereich durch blaue Frames und einen durchgezogenen Pfeil vom Anfangs-Schlüsselframe zum End-Schlüsselframe angezeigt (siehe Abbildung 16.19).



**Abbildung 16.19: Motion Tween**

8. Testen Sie den Motion Tween durch Drücken der (Return)-Taste.
9. Vervollständigen Sie die Animation durch einen zweiten Motion Tween, der den Ball wieder nach oben hüpfen lässt.

10. Speichern und exportieren Sie die Animation (siehe Abschnitt 16.5.1).

## Shape Tween (Morphing)

Flash kann neben Bewegungen auch Formen animieren (Morphing). In diesem Falle gehen Sie grundsätzlich genauso vor, wie beim Motion Tween. Statt aber im End- Schlüsselframe das zu animierende Objekt zu verschieben, verändern Sie seine Form (oder zeichnen eine neue Form). Außerdem müssen Sie den Befehl **Window/Panels/ Fame** aufrufen und in dem erscheinenden Dialogfeld unter **Tweening** die Option **Shape** auswählen.

## Komplexere Animationen mit Hintergründen und Sprites

Zum Schluss wollen wir noch eine etwas komplexere Animation erstellen. Sie soll einen von einer Fliege umschwirrten Cartoon-Kopf vor einem Sonnenuntergang zeigen. Sonnenuntergang, Kopf und Fliege werden wir jeweils auf eigenen Ebenen anlegen - so bleibt die Animation übersichtlich und wir können die Ebenen für sich bearbeiten. Für die Animation der Fliege werden wir eine weitere Variante des Motion Tweens kennen lernen: die Bewegung entlang eines Laufpfades.

### Neue Animation beginnen

1. Schließen Sie die alte Animation, indem Sie den Befehl **File/Close** aufrufen.
2. Beginnen Sie eine neue Animation mit **File/New**.
3. Richten Sie die Animation wie im Abschnitt 16.5.1 ein. Setzen Sie die Leinwandgröße diesmal auf 300 mal 150 Pixel.
4. Speichern Sie die neue Animation unter dem Namen *kopf fla*.

### Bilder importieren

Als Hintergrund unserer Animation wollen wir das Photo eines Sonnenuntergangs verwenden. Die Bilddatei, *himmel.jpg*, finden Sie auf der Buch-CD.

5. Markieren Sie in der Zeitleiste den ersten Frame.
6. Rufen Sie den Befehl **File/Import** auf und wählen Sie im erscheinenden Dialogfenster die Datei *himmel.jpg* aus.
7. Platzieren Sie das importierte Bild mittig im Frame.

### Statische Inhalte

Das Hintergrundbild soll die gesamte Animation über unverändert bleiben. Sagen wir unsere Animation soll 15 Frames umfassen. Wir könnten jetzt 14 weitere Frames anlegen und das Hintergrundbild in jeden dieser Frames neu importieren. Wir können uns die Arbeit aber auch vereinfachen.

8. Klicken Sie in der Zeitleiste mit der rechten Maustaste in den - sagen wir - fünfzehnten Frame und wählen Sie den Befehl **Insert Frame** aus (nicht **Insert Keyframe** wie beim Tweening)!

In der Zeitleiste erscheinen die Frames jetzt grau, was bedeutet, dass sie alle

den gleichen Inhalt haben wie der vorangehende Schlüsselframe (mit dem Punkt). Der letzte Frame der Reihe enthält ein weißes Kästchen.



Abbildung 16.20: Unveränderter Frameinhalt

## Neue Ebene anlegen

Über diesen Hintergrund wollen wir den Cartoon-Kopf aus *kopf.gif* legen. Dazu verwenden wir eine zweite Ebene.

9. Rufen Sie den Befehl **Insert/Layer** auf.

Die neue Ebene wird in der Zeitleiste über der alten Ebene eingeblendet.

Da wir an der alten Ebene nichts mehr verändern wollen, ist es sinnvoll, die alte Ebene vor unabsichtlichen Änderungen zu schützen. Klicken Sie in der Layer 1 auf den Punkt unter dem Sicherheitsschloss.

10. Markieren Sie den ersten Frame in der 2. Ebene (Layer 2).

11. Rufen Sie den Befehl **File/Import** auf, um das Bild des Kopfes (*kopf.gif*) zu importieren.

Verschieben Sie den Kopf irgendwo an den unteren Rand der Leinwand.

Da wir auch an dieser Ebene nichts mehr verändern wollen, ist es sinnvoll, die Ebene vor unabsichtlichen Änderungen zu schützen. Klicken Sie in der Layer 2

auf den Punkt unter dem Sicherheitsschloss.



Abbildung 16.21: Animation mit zwei Ebenen



*Beachten Sie, dass Flash den importierten Kopf automatisch in alle Frames der Animation kopiert. Beachten Sie des Weiteren, dass der Hintergrund nur durchscheint, wenn das GIF-Bild des Kopfes mit transparenter Hintergrundfarbe abgespeichert ist.*

## Ebenen ausblenden

12. Legen Sie mit **Insert/Layer** eine dritte Ebene an.
13. Markieren Sie den ersten Frame in der 3. Ebene (Layer 3).
14. Rufen Sie den Befehl **File/Import** auf, um das Bild der Fliege (*fliege.gif*) zu importieren.

Verschieben Sie die Fliege zum Ohr des Kopfes.

Die schwarze Fliege sieht man nur schlecht vor dem Hintergrund des Sonnenuntergangs. Da der Hintergrund für die weitere Bearbeitung der Animation nicht wichtig ist, blenden wir ihn aus.

15. Klicken Sie einfach in das Farbfeld der 1. Ebene (Layer 1). Das Farbfeld wechselt von ausgefüllt zu umrahmt und der Inhalt der Ebene wird in den Frames ausgeblendet.

## Motion Tween entlang eines Laufpfades

Als Erstes machen wir aus der statischen Animation der Fliege einen Motion Tween.

16. Markieren Sie die Fliege im Frame und rufen Sie den Befehl **Insert/Create Motion Tween** auf.
17. Klicken Sie mit der rechten Maustaste in der Zeitleiste in den letzten (fünfzehnten) Frame und wählen Sie den Befehl **Insert Keyframe** aus.
18. Verschieben Sie die Fliege im letzten Frame zur Endposition der Animation (unter der Nase).

Die Fliege soll aber nicht auf einer geraden Linie zwischen Start- und Endpunkt verschoben werden, sondern auf einer Wellenlinie. Diese Wellenlinie, den Laufpfad des Motion Tweens, müssen wir als eigene Ebene einrichten.

19. Markieren Sie die Ebene der Fliege.
20. Klicken Sie am unteren Rand der Zeitleiste auf die Schaltfläche **Add Guide Layer**.
21. Zeichnen Sie jetzt mit dem Stift-Werkzeug den Laufpfad ein.

Aktivieren Sie zuvor die Zwiebelhüllenansicht und dehnen sie diese soweit aus, dass sowohl Start- wie Endpunkt der Fliege eingeschlossen sind (siehe oben)  
Dies ist zwar nicht unbedingt notwendig, aber es erleichtert das Einzeichnen des Laufpfades, wenn Start- und Endpunkt zu sehen sind.

Wählen Sie dann in der Tools-Leiste das Stift-Werkzeug (Pencil) und zeichnen Sie den Laufpfad ein.

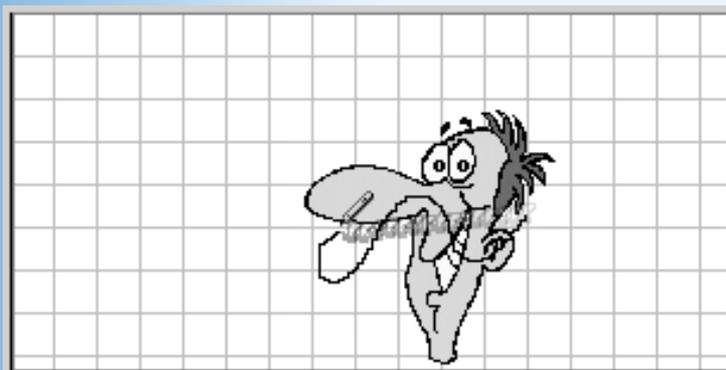


Abbildung 16.22: Einzeichnen des Laufpfades



*Start- und Enddarstellung des zu bewegenden Objekts müssen direkt über Beginn*

und Ende des Laufpfads liegen. Wechseln Sie gegebenenfalls in die Ebene der Fliege und positionieren Sie in Start- und Ende-Frame das Kreuz des Fliegen-Objekts über dem Laufpfad.

## Bewegtes Objekt in Pfadrichtung drehen

Jetzt wollen wir nur noch erreichen, dass unsere Fliege sich so dreht, wie sie fliegt.

22. Markieren Sie die Ebene der Fliege und in der Ebene das Fliegen-Objekt selbst.
23. Rufen Sie dann den Befehl **Modify/Instance** auf.
24. Wechseln Sie im Dialogfeld zur Registerkarte **Frame**. Aktivieren Sie dort die Option **Orient to Path** und wählen Sie für **Rotate** den Eintrag **None**.

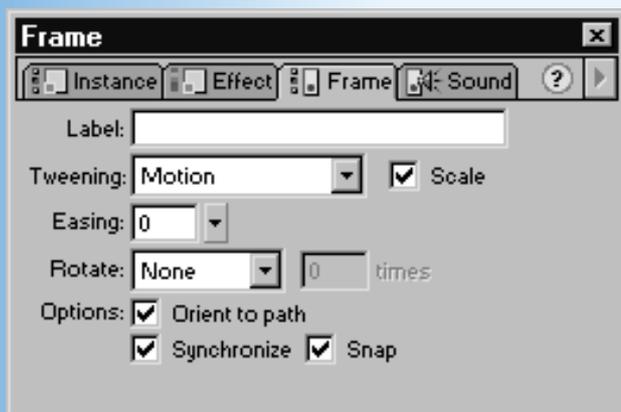


Abbildung 16.23: Objekte in Pfadrichtung drehen

25. Speichern und veröffentlichen (publish) Sie die Animation.
26. Laden Sie die von Flash erzeugte HTML-Seite in Ihren Browser.

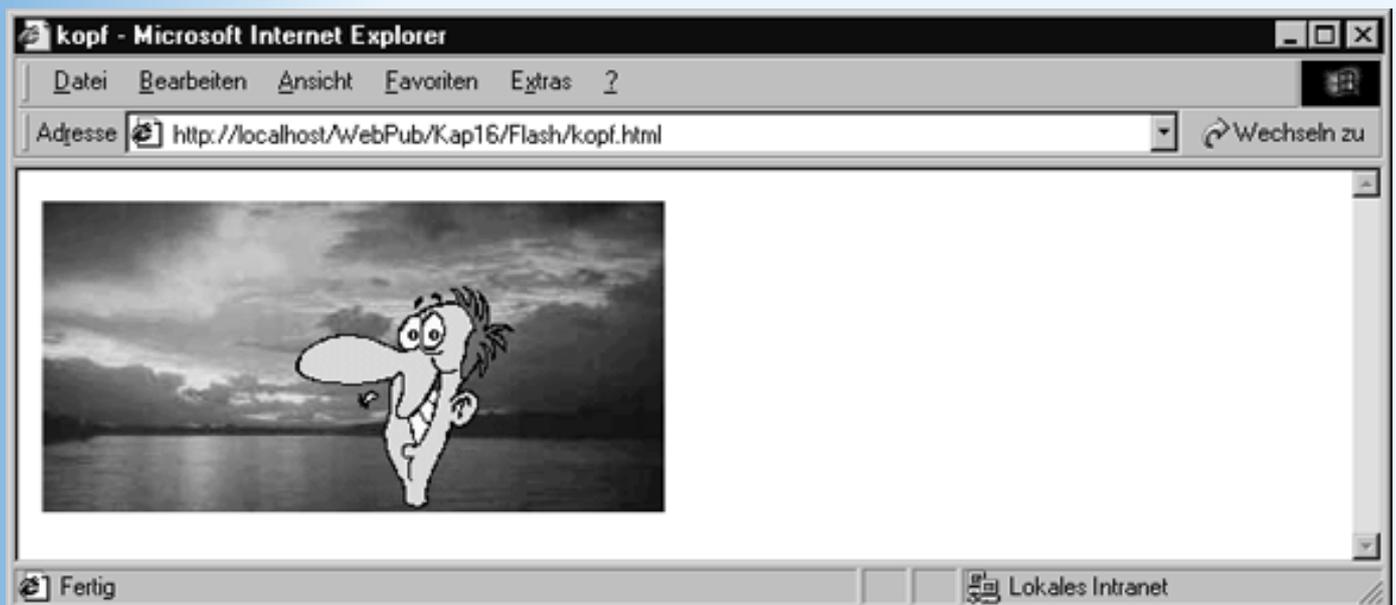


Abbildung 16.24: Animation mit drei Ebenen und Laufpfad

Damit beenden wir unsere Einführung in die Erstellung von Flash-Animationen. Viele interessante Optionen und Möglichkeiten der Flash-Software konnten wir nicht ansprechen, beispielsweise

das Abbremsen von Bewegungen oder das Einbinden von Sound. Wenn Sie mehr über Flash erfahren wollen, schauen Sie sich die Dokumentation an, die der Software beiliegt oder suchen Sie nach geeigneter Fachliteratur.

## 16.6 Zusammenfassung

Nach dem anstrengenden Java-Kapitel am gestrigen Tag haben wir uns heute ein wenig Entspannung gegönnt und ein Thema angeschnitten, für das sich wohl jeder Webdesigner begeistern kann: der Erstellung von Animationen.

Nach einer ganz kurzen Einführung in einige Grundbegriffe der Animationstechnik sind wir direkt in media res gegangen und haben uns angeschaut, wie man mit Hilfe von

- GIFs
- Java
- JavaScript und
- Flash

Animationen für Webseiten erstellen kann.

Wir haben Ihnen Beispiele für Bildwechsler, Zeichentrickanimationen und Laufschriften gezeigt und Sie in die Arbeit mit dem Ulead GIF-Animator sowie der Flash-Software von Macromedia eingeführt.

## 16.7 Fragen und Antworten

**Frage:**

**Kann man mit Java auch Animationen mit Bildern schreiben?**

*Antwort:*

*Ja, allerdings muss man dazu wissen, wie man Bilder in Applets lädt, was nicht ganz trivial ist (siehe entsprechende Fachliteratur).*

**Frage:**

**Ist Flash auch für den Macintosh verfügbar?**

*Antwort:*

*Ja, nur für UNIX/Linux-Plattformen gibt es derzeit noch keine Flash-Software.*

## 16.8 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

1. Nennen Sie eine Software zur Erstellung von GIF-Animationen, die für alle Plattformen verfügbar ist!
2. Wie kann man erreichen, dass eine JavaScript-Funktion in regelmäßigen Zeitabständen immer wieder aufgerufen wird?
3. Wie kann man Java-Applets threadfähig machen?
4. Welche Flash-Animationstechniken haben Sie kennen gelernt?
5. Wie bindet Flash seine Animationen in HTML-Code ein?

## Übungen

Zu dem heutigen Tag gibt es keine vorgeschriebenen Übungen. Nutzen Sie die Zeit, um mit Hilfe irgendeiner der an diesem Tag beschriebenen Techniken eine eigene Animation zu implementieren.

---

❖ Kapitel Inhalt Index **SAMS** ❖ Top Kapitel ❖

---

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

## Tag 17

### CGI und Perl

Heute kommen wir auf ein Thema zu sprechen, dass wir bereits an mehreren Stellen im Buch angekündigt haben: CGI und die Implementierung serverseitiger Programme zur Unterstützung und dynamischen Erzeugung von Webseiten.

Die Themen im Einzelnen:

- Vorstellung des CGI-Konzepts
- Einrichtung des Webservers
- Einrichtung des Perl-Interpreters
- Einführung in die Programmiersprache Perl
- Dynamisches Erzeugen von Webseiten
- Implementierung serverseitiger ImageMaps
- Serverseitige Bearbeitung von Formulareingaben
- »Geheime« Daten via verborgenen Feldern übertragen
- Implementierung eines Gästebuchs
- Implementierung einer Umfrage (Übung 1)

### 17.1 Wie funktioniert CGI?

CGI steht für Common Gateway Interface - eine Schnittstellenspezifikation, die festlegt, wie der Server CGI-Programme aufruft, Eingaben vom Browser an das CGI-Programm weiterleitet und die Ausgabe des CGI-Programms an den Browser zurückliefert.

Was aber ist ein CGI-Programm? CGI-Programme sind ganz normale Programme, die in praktisch jeder beliebigen echten Programmiersprache geschrieben sein können. Das Einzige, was CGI-Programme von anderen Programmen unterscheidet (was sie also überhaupt erst zu CGI-Programmen macht), ist, dass sie wissen, wie der Webserver Daten vom Browser an sie weiterreicht und dass sie gegebenenfalls Ausgaben erzeugen, die der Webserver an den Browser zurückgibt.

#### Das CGI-Modell

Im einfachsten Fall läuft die CGI-Kommunikation wie folgt ab:

Ein Webautor möchte, dass beim Klick auf einen bestimmten Hyperlink (oder den Abschicken-Schalter eines Formulars) ein CGI-Programm aufgerufen wird, das er auf dem Webserver installiert hat. Also weist er dem href-Attribut des Hyperlinks statt des URLs eines HTML-Dokuments den URL des CGI-Programms zu.

```
CGI-Programm
```

Die Webseite mit diesem Hyperlink wird nun von einem Websurfer geladen und im Browser des Websurfers angezeigt. Irgendwann klickt der Websurfer auf den Hyperlink.

1. Der Browser schickt jetzt den URL des CGI-Programms an den Webserver.

```
http://webserver/cgi-bin/programm.pl
```

2. Der Webserver nimmt die Anforderung entgegen, sucht auf seiner Festplatte das CGI- Programm *programm.pl*

und ruft es auf.

3. Das CGI-Programm erzeugt als Ausgabe eine HTML-Datei, die es an den Server zurückgibt. Dazu gibt das CGI-Programm einfach den HTML-Code der Datei Zeile für Zeile auf der Konsole aus.
4. Der Webserver schickt die Ausgabe des Programms an den Browser.

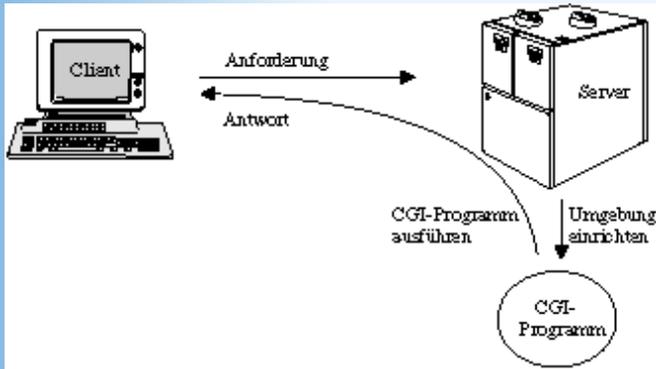


Abbildung 17.1: CGI-Kommunikation

## Was kann man mit CGI machen?

Mit CGI können wir:

- Webseiten dynamisch erstellen lassen
- Formulareingaben verarbeiten
- Kleine Dienstprogramme schreiben - beispielsweise für Passwortabfragen, Zugriffszähler, etc.

Der Vorteil der CGI-Programme ist, dass Sie im Gegensatz beispielsweise zu JavaScript- Code auf der Serverseite ausgeführt werden, das heißt, Sie können Daten, die vom Browser kommen, auf dem Webserver abspeichern (beispielsweise Kundenbestellungen), oder umgekehrt, Daten, die auf dem Webserver verfügbar sind (beispielsweise Datenbankinhalte) ad hoc in Webseiten einbauen.

Zuvor aber muss der Webserver so eingerichtet werden, dass er die von uns geschriebenen CGI-Programme verarbeiten kann.

## Einrichtung des Webservers

Als Programmiersprache für die Erstellung unserer CGI-Programme haben wir uns für Perl entschieden. Um die Beispiele dieses Tages nachvollziehen zu können, müssen Sie also einen Perl-Interpreter auf Ihrem System installieren. Später, bei der Konfiguration des Webservers für die Verarbeitung von Perl-CGI-Programmen werden wir dem Webserver mitteilen, wo er den Perl-Interpreter<sup>1</sup> finden kann.

## Perl installieren

Wenn Sie mit Unix/Linux arbeiten, müssen Sie wahrscheinlich nichts weiter tun, da Perl vermutlich bereits auf Ihrem System installiert ist. Rufen Sie einfach ein Konsolenfenster (je nach verwendetem Windowing-Manager auch Terminal genannt) auf, und tippen Sie am Prompt

```
perl -v
```

ein. Wenn Perl auf Ihrem System installiert ist, wird Ihnen daraufhin die Kennnummer der installierten Version angezeigt. Zum Zeitpunkt der Drucklegung dieses Buches war 5.6.0 aktuell. Jede Version, deren Kennnummer mit 5 beginnt, sollte aber für den Einstieg vollkommen ausreichend sein. (Wenn Sie eine aktuellere Perl-Version installieren wollen, surfen Sie zur Website Ihres Linux-Vertreibers, zu [www.activestate.com/ActivePerl](http://www.activestate.com/ActivePerl) oder zu [www.perl.com](http://www.perl.com) und laden Sie sich von dort kostenfrei die aktuelle Version herunter.)

Wenn Sie mit Windows arbeiten, surfen Sie zur Website [www.activestate.com/ActivePerl](http://www.activestate.com/ActivePerl) und laden Sie sich von dort die Binärversion (Binary Distribution) für Windows (Win32) herunter. Folgen Sie den Anleitungen zur Installation. (An sich brauchen Sie nur auf die heruntergeladene Installationsdatei doppelklicken. Windows 95/98-Anwender müssen

aber eventuell zuvor noch *instmsi.exe* installieren. Lesen Sie also auf jeden Fall die Installationsanweisungen auf der ActiveState-Website.)

## Webserver für CGI mit Perl konfigurieren

Die vermutlich schwierigste Aufgabe bei der CGI-Programmierung ist die Einrichtung des Webserver. Dieser muss so konfiguriert werden, dass er die Ausführung von Programmen aus bestimmten Verzeichnissen gestattet und dass er zur Ausführung von Perl-Programmen den Perl-Interpreter heranzieht.

Sofern Sie noch keinen Webserver eingerichtet haben, müssen Sie dies jetzt nachholen. In Kapitel 2 und Anhang A ist die Einrichtung und Konfiguration gängiger Webserver beschrieben (Apache, PWS, IIS, OmniHTTPd).

Danach müssen Sie den Webserver so konfigurieren, dass er die Ausführung von Perl-CGI- Programmen unterstützt.



*Die in diesem Abschnitt angegebenen Pfadangaben sind größtenteils Standardwerte, die bei der Installation verändert werden können. Es ist also durchaus möglich, dass auf Ihrem Rechner abweichende Pfade und Verzeichnisse verwendet werden.*

- Für Ihren Apache-Server wurde bereits ein Verzeichnis *cgi-bin* eingerichtet (je nach Version unter */home/httpd/cgi-bin* oder */usr/local/apache/cgi-bin*). Meist ist der Apache- Server so eingerichtet, dass Sie Ihre Skripte nur noch in dieses Verzeichnis kopieren müssen. Unter Umständen müssen Sie aber noch ein Skript-Alias einrichten, das Zugriffe auf *http://servername/cgi-bin* zu dem **cgi-bin**-Verzeichnis umleitet (das ja kein Unterverzeichnis des Dokumentenverzeichnisses ist). Loggen Sie sich dann als Root ein, laden Sie die Konfigurationsdatei *httpd.conf* und erweitern Sie diese um folgenden Eintrag:

```
ScriptAlias /cgi-bin/ /home/httpd/cgi-bin/2
<Directory /home/httpd/cgi-bin>
 AllowOverride None
 Options None
</Directory>
```

Starten Sie den Server danach neu (je nach Version */etc/rc.d/init.d/httpd restart* oder */usr/local/apache/bin/apachectl restart*).

- Wie Sie den Personal Web Server oder den IIS einrichten, hängt von der Server- Version und dem verwendeten Betriebssystem ab. Ausführlichere Informationen hierzu finden Sie in der Dokumentation zu Ihrem Server. Die Hilfedateien werden standardmäßig im virtuellen Verzeichnis *IISHelp* abgespeichert, das meist dem Windows-Unterverzeichnis *Help/IIS* entspricht. Halten Sie dort zum Beispiel nach der Datei *iicacgia.htm* Ausschau. (Sie können die Hilfe zum Server auch über Ihren Browser, *http://localhost/iishelp*, aufrufen.)

Falls Sie ActivePerl installiert haben, können Sie auch in der HTML-Dokumentation nachschlagen (*C:/Perl-Verzeichnis/html/index.html*).

- Wenn Sie mit dem OmniHttpd arbeiten, klicken Sie in der Windows-Taskleiste mit der rechten Maustaste auf das Symbol des Servers und rufen Sie den Befehl **Properties** auf. In älteren Versionen wechseln Sie daraufhin zum Register »**Advanced**«, aktivieren dort die Option **Enable Perl CGI Support**, tragen im Eingabefeld **Perl CGI Command Line** den Pfad zu Ihrem Perl-Interpreter ein (beispielsweise *C:\perl\bin\perl.exe*) und starten danach den Server neu. In der aktuellen Version (2.0.7) klicken Sie im **Configuration**-Dialog auf den Schalter **Web Server Global Settings**, wechseln zur Registerkarte **External**, markieren nacheinander die Einträge für *.pl* und *.plx* und passen jeweils den Pfad so an, dass er zu dem auf Ihrem Rechner installierten Perl-Interpreter führt.

## CGI-Kommunikation testen

Bevor wir prüfen, ob unser Webserver korrekt für die Verarbeitung von Perl-CGI- Programmen eingerichtet ist, sollten wir uns vergewissern, ob überhaupt der Perl- Interpreter korrekt installiert ist.

## Perl-Installation testen

1. Rufen Sie einen Texteditor auf (Notepad, vi, etc.) und tippen Sie folgenden Quelltext ein:

### Listing 17.1: hallo.pl

```
#!/usr/bin/perl -w
print("Hallo Welt!\n");
```

2. Speichern Sie die Datei als *hallo.pl*.
3. Führen Sie das Skript aus. Rufen Sie dazu vom Prompt der Konsole (unter Windows ist dies die MSDOS-Eingabeaufforderung, die über **Start/Programme** aufgerufen wird) den Perl-Interpreter auf und übergeben Sie ihm das Perl-Skript:

```
perl hallo.pl
```

Danach sollte als Ausgabe

Hallo Welt!

erscheinen.



*Wenn Sie mit Windows NT arbeiten und die Dateierweiterung .pl mit dem Perl-Interpreter verbunden haben, können Sie die Skripte auch direkt mit ihrem Namen aufrufen und ausführen lassen. Auch unter Unix/Linux genügt der Aufruf des Skriptnamens, der zugehörige Interpreter wird dann der auskommentierten ersten Zeile des Skripts entnommen. Voraussetzung ist allerdings, dass das Skript ausführbar ist:*

```
> chmod +x hallo.pl
> hallo.pl3
```

## Perl/CGI-Konfiguration des Webserver testen

Testen Sie die Ausführung von Perl-CGI-Skripten auf dem Server. Setzen Sie dazu das folgende Perl-Skript auf und speichern Sie die Datei im **cgi-bin**-Verzeichnis Ihres Servers. (Führen Sie das Skript zur Probe einmal aus, um sich zu vergewissern, dass es syntaktisch korrekt ist.)

### Listing 17.2: testecgi.pl

```
#!/usr/bin/perl -w
print "Content-type: text/html\n\n";
print "<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01//EN\" \n";
print " http://www.w3.org/TR/html4/strict.dtd\"> \n";
print "<html>\n";
print "<head>\n";
print "<title>CGI-Testprogramm</title>\n";
print "</head>\n";
print "<body>\n";
print "<p>Hallo vom CGI-Programm!</p>\n";
print "</body></html>\n";
```

Achten Sie darauf, dass die Datei vom Webserver ausgeführt werden kann.

- Wenn Sie unter Linux mit Apache arbeiten, ändern Sie die Zugriffsrechte für die Datei mit **chmod 755 testcgi.pl**.
- Wenn Sie mit dem PWS oder IIS arbeiten, müssen Sie bereits bei der Einrichtung des cgi-bin-Verzeichnisses darauf achten, dass dieses das Ausführen der darin enthaltenen Dateien erlaubt.
- Wenn Sie mit dem OmniHttpd arbeiten, können Sie alternativ auch die Webseite **default.htm** aufrufen, die einen Test-Link für Perl enthält.

Rufen Sie dann das Skript über Ihren Browser auf, indem Sie einfach statt dem URL einer Webseite den URL des Perl-Programms eingeben (siehe Abbildung 17.2).



Abbildung 17.2: Ausgabe des Testskriptes im Browser



*Je nachdem, wie das Dokumentenverzeichnis Ihres lokalen Webserver heißt und in welchen Verzeichnissen Sie Ihre Webseiten abgelegt haben, müssen Sie die URL-Angaben in den Skripten und Webseiten dieses Kapitels ändern.*

## 17.2 Perl

Perl ist, wie bereits gesagt, nur eine von vielen Programmiersprachen, die man für die Erstellung von CGI-Programmen verwenden kann. Perl wird traditionell aber häufig zur CGI-Programmierung herangezogen - und zwar aus gutem Grund:

- Perl ist auf UNIX/Linux-Rechner weit verbreitet.
- Perl ist frei verfügbar (auch für Windows, siehe Abschnitt 17.1.2).
- Perl ist zwar etwas kryptisch, aber ansonsten nicht schwer zu erlernen.
- Perl verfügt über Eigenschaften, die gerade die dynamische Erstellung von Webseiten stark vereinfachen.
- Perl verfügt über eine umfangreiche, öffentliche Bibliothek von nützlichen Programm-Modulen (das CPAN), auf die jeder Perl-Programmierer zur Lösung von Standardaufgaben zurückgreifen kann.

Der nachfolgende Programmierkurs soll Sie in die Programmierung mit Perl einführen und Sie mit den wichtigsten syntaktischen Elementen bekannt machen. Der Kurs ist recht stringent aufgebaut und erinnert streckenweise mehr an eine Referenz als an einen Lehrkurs. Dies hat seine Gründe. Zum einem sind Ihnen viele der hier vorgestellten Konzepte und Konstrukte bereits von JavaScript her bekannt und bedürfen daher keiner weiteren Erläuterung, zum anderen interessiert uns an Perl vor allem die Verwendung als Sprache zur CGI-Programmierung und so wollen wir den Syntax-Teil möglichst schnell abschließen, um mehr Zeit für die CGI-Programmierung zu haben. Schließlich kommt noch hinzu, dass Perl eine so umfangreiche Sprache mit so vielen Facetten ist, dass man ihr unmöglich im Rahmen dieses Buches gerecht werden könnte. Wer mehr über Perl erfahren möchte, der sei an entsprechende Fachbücher wie »Jetzt lerne ich Perl« oder »Programming Perl« verwiesen (siehe Anhang E).

# Ein erstes Perl-Programm

Unser erstes Perl-Programm haben wir bereits in Abschnitt 17.1.3, bei der Kontrolle der Perl-Installation aufgesetzt.

## Listing 17.3: hallo.pl

```
#!/usr/bin/perl -w
print("Hallo Welt!\n");
```

Die erste Zeile in diesem Programm ist ein Kommentar.

## Kommentare

Kommentare beginnen mit einem #-Zeichen und reichen bis zum Ende der Zeile. Sie werden vom Perl-Interpreter ignoriert - sie dienen lediglich zur Dokumentation des Quelltextes.

## Der Shebang-Kommentar

Der Kommentar in der ersten Zeile erfüllt allerdings unter UNIX/Linux einen besonderen Zweck: er teilt UNIX/Linux mit, mit welchem Interpreter das Skript auszuführen ist; wobei auch - wie man sieht - Kommandozeilenargumente an den Interpreter übergeben werden können (-w). Wenn Sie also unter UNIX/Linux arbeiten und Ihr Perl-Interpreter in einem anderen Verzeichnis als /usr/bin/ steht, ändern Sie die erste Zeile entsprechend ab. Webserver der Microsoft Windows-Plattform ignorieren meist die Pfadangabe und werten nur die Optionen für den Interpreter aus.



*Die Amerikaner bezeichnen die anfängliche Kommentarzeile als Shebang, da sie mit einem # (im Amerikanischen als sharp bezeichnet) und einem ! (dem bang) beginnt.*

Unter dem Kommentar folgt eine Leerzeile, die der besseren Lesbarkeit des Skripts dient.

Der eigentliche Code des Skripts steht in der dritten Zeile. Diese Zeile enthält eine Anweisung, was an dem abschließenden Semikolon zu erkennen ist. Die Anweisung ruft die Funktion print auf.

## Ausgaben

print ist der Name einer Funktion, die in den Perl-Bibliotheken vordefiniert ist und die dazu dient, Text auszugeben. Der auszugebende Text folgt direkt hinter dem Namen der Funktion und ist in doppelte Anführungszeichen gefasst, woran der Perl-Interpreter erkennt, dass es sich um Text handelt, den das Programm (und nicht der Interpreter) verarbeitet. Die Anweisung aus der dritten Zeile besteht also aus dem Aufruf der Funktion print, der als auszugebender Text der String "Hallo Welt!\n" übergeben wird.



*In JavaScript müssen die Argumente, die man einer Funktion beim Aufruf übergibt, immer in runden Klammern auf den Funktionsnamen folgen. In Perl können Sie sich die runden Klammern sparen, wenn auf den Funktionsnamen nur noch Argumente für die Funktion folgen (siehe beispielsweise Listing 17.2).*

## Variablen und Konstanten

Perl kennt - im Gegensatz zu JavaScript - verschiedene Variablentypen: Skalare, Listen, Hashes, Referenzen und Datei-Handles. Konzentrieren wir uns zuerst auf die Skalare.

## Skalare Variablen definieren

Skalare Variablennamen beginnen in Perl immer mit einem \$. Auf das Präfix folgt der eigentliche Variablenname. Dieser Variablenname darf aus Buchstaben, Ziffern und Unterstrichen bestehen, darf aber nicht mit einer Ziffer anfangen. Des Weiteren muss er eindeutig sein (wobei Perl zwischen Groß- und Kleinschreibung unterscheidet).

Gültige Variablennamen wären also

```
$meineVariable
$meine_variable
$_meineVariable
$meine3teVariable
```

Nicht gültig wären

```
$3teVariable # beginnt mit einer Ziffer
$meine%Variable # enthält ungültiges Zeichen %
```

Wie in JavaScript brauchen Variablen nicht vorab deklariert zu werden.

```
#!/usr/bin/perl -w
$zahl1 = 100;
$zahl2 = 12;
print $zahl1 + $zahl2; # Ausgabe: 112
```

Diese Verfahrensweise ist allerdings recht fehleranfällig. Vertippt man sich bei einem Variablennamen (schreibt man in obigem Code beispielsweise `print $zahl1 + $zahl2;`), erzeugt der Perl-Interpreter unbemerkt eine neue Variable `$zahl2` mit undefiniertem Wert und rechnet mit dieser weiter - was natürlich zu falschen Ergebnissen führt. Um dies zu vermeiden, sollten Sie Ihre Perl-Programme so aufsetzen, dass der Interpreter nur vorab definierte Variablennamen akzeptiert.

1. Aktivieren Sie das Perl-Pragma `use strict;`
2. Deklarieren Sie alle Variablen mit dem Schlüsselwort `my`

### Listing 17.4: variablen.pl

```
#!/usr/bin/perl -w
use strict;
my $zahl1 = 100;
my $zahl2 = 12;
print $zahl1 + $zahl2;
```

## Konstanten

Zahlenkonstanten können in Perl auf verschiedene Weise angegeben werden:

```
100 # als Ganzzahl
333.33 # als Gleitkommazahl
3.1e2 # in Exponentialschreibweise
0x12E # als Hexadezimalzahl
```

Na, das erinnert doch sehr an JavaScript. Auch die String-Konstanten scheinen auf den ersten Blick der von Javascript gewohnten Syntax zu folgen:

```
"Ich bin ein String"
'Ich bin auch ein String'
```

Allerdings gibt es in Perl einen wichtigen Unterschied zwischen Strings in einfachen oder doppelten

Anführungszeichen.

In Perl kann man nämlich Variablennamen direkt in Strings einbauen:

```
"100 + 12 = $summe"
```

Steht der String in doppelten Anführungszeichen ersetzt der Perl-Interpreter den Variablennamen durch den Wert der Variablen. Steht der in einfachen Anführungszeichen, gibt der Interpreter den Variablennamen wie eine ganz normale Zeichenfolge aus.

### Listing 17.5: konstanten.pl

```
#!/usr/bin/perl -w
use strict;
my $summe = 100.33 + 12;
print "100.33 + 12 = $summe";
```



*In Strings mit doppelten Anführungszeichen können Sie auch Sonderzeichen - wie `\n` (Zeilenumbruch), `\t` (Tabulator), `\"` (doppeltes Anführungszeichen) und andere - einbauen.*

Eine weitere Spezialität von Perl sind die HERE-Texte.

### HERE-Texte

Dabei wird der auszugebende Text, der ruhig mehrere Zeilen umfassen kann (und sollte) zwischen zwei frei wählbaren Bezeichnern definiert und mit Hilfe des `<<`-Operators in einer Variablen gespeichert

### Listing 17.6: here.pl

```
#!/usr/bin/perl -w
$gedicht = <<HERE_DOUGLAS;
 Archibald Douglas
 von
 Theodor Fontane
 Ich habe es getragen sieben Jahr,
 Und ich kann es nicht tragen mehr!
 Wo immer die Welt am schönsten war,
 Da war sie öd und leer.
 ...
HERE_DOUGLAS
print $gedicht;
```

- Eingeleitet wird ein HERE-Text mit dem `<<`-Operator gefolgt von dem HERE- Bezeichner, den Sie selbst wählen. Zwischen dem Operator und dem Bezeichner darf kein Leerzeichen stehen. Wenn Sie den HERE-Bezeichner in Großbuchstaben schreiben und mit dem Präfix HERE beginnen (was beides nicht erforderlich ist), sind Anfang und Ende des HERE-Textes gut zu erkennen.
- Der eigentliche Text beginnt in der nachfolgenden Zeile und wird so ausgegeben, wie er im Quelltext steht. Variablennamen werden expandiert (durch ihren Wert ersetzt).
- Das Ende des HERE-Textes wird durch die Wiederholung des HERE-Bezeichners angezeigt. Der HERE-Bezeichner muss dabei allein und am Anfang der Zeile stehen.



Später, wenn wir mit Perl-CGI-Programmen dynamisch Webseiten erzeugen, wird uns diese Technik sehr zugute kommen!

## Aus- und Eingabe

Wie man mit Hilfe der print-Funktion Strings und Werte von Variablen ausgibt, konnten Sie bereits in den vorangehenden Beispielen sehen. Als Ergänzung zu dieser Funktion kennt Perl noch eine weitere Ausgabefunktion, printf, mit der man das Ausgabeformat für in den String eingebaute Variableninhalte bestimmen kann.

### Formatierte Ausgabe mit printf

Betrachten wir folgenden Code:

```
#!/usr/bin/perl -w
use strict;
my $zahl1 = 100;
my $zahl2 = 12;
my $ergebnis;
$ergebnis = $zahl1 / $zahl2;
print "$zahl1 / $zahl2 = $ergebnis ";
```

Wenn Sie dieses Programm ausführen, erhalten Sie folgende Ausgabe:

```
100 / 12 = 8.333333333333333
```

Die vielen Nachkommastellen in dieser Ausgabe sind etwas unschön. Mit Hilfe von printf können wir festlegen, dass nicht mehr als, sagen wir, 3 Nachkommastellen ausgegeben werden.

Der Trick von printf ist, dass man statt der Variablennamen Platzhalter in den String einfügt (beispielsweise %s für einen String, %d für eine Ganzzahl, %f für eine Gleitkommazahl). Im Gegensatz zu den Variablennamen kann man für die Platzhalter angeben, wie die Werte, die sie repräsentieren, formatiert werden sollen.

Formatierung	Effekt	Beispiel
+	Zahlen immer mit Vorzeichen	"%+d"
-	Linksbündige Ausrichtung	"%-d"
b	Angabe der Feldbreite der Ausgabe	"%5d"
.n	Angabe der Nachkommastellen für Gleitkommazahlen	"%.3f"

**Tabelle 17.1: Ausgesuchte Formatflags für printf**

Die Variablen, deren Werte an der Stelle der Platzhalter ausgegeben werden sollen, werden im Anschluss an den auszugebenden String aufgeführt (in der Reihenfolge, in der die Werte den Platzhaltern zugeteilt werden sollen).

#### Listing 17.7: printf.pl

```
#!/usr/bin/perl -w
use strict;
my $zahl1 = 100;
my $zahl2 = 12;
my $ergebnis;
```

```
$ergebnis = $zahl1 / $zahl2;
printf "%d / %d = %.3f ", $zahl1, $zahl2, $ergebnis;
```

## Umlaute auf der Windows-Konsole

Wenn Sie Text mit deutschen Umlauten auf die Windows-Konsole (MSDOS-Eingabeaufforderung) ausgeben, sehen Sie statt der Umlaute nur grafische Sonderzeichen. Dies liegt daran, dass die Windows-Konsole einen anderen Zeichensatz (OEM-Zeichensatz) verwendet als Windows. Sie können diese Manko beheben, indem Sie statt der Zeichen den ASCII-Code angeben:

```
print "Umlaute \x84 \x94 \x81 \n";
```

## Daten über die Konsole einlesen

Mit Perl kann man Daten über die Tastatur einlesen. Für CGI-Programme ist diese Technik zwar uninteressant (weil es keinen Anwender gibt, der die Daten eintippen könnte), aber sie ermöglicht es uns, die Beispielprogramme in diesem Abschnitt etwas interessanter zu gestalten.

In Perl-Programmen wird die Tastatur durch das vordefinierte Handle STDIN repräsentiert. In Verbindung mit dem Einleseoperator <> kann man über diesen Handle Daten von der Tastatur einlesen.

```
my $meineVar;
$meineVar = <STDIN>;
chomp($meineVar);
```

Zuerst definiert man eine Variable (hier \$meineVar).

Im nächsten Schritt werden die Daten eingelesen. Dazu wendet man auf den Datei- Handle den Einleseoperator <> an, was dann wie folgt aussieht: <STDIN>. Der Einleseoperator liest so lange Daten aus der Datenquelle, bis er auf ein bestimmtes Trennzeichen trifft. Per Voreinstellung ist dieses Trennzeichen das Neue-Zeile-Zeichen. Die Konstruktion <STDIN> liest also so lange Zeichen über die Tastatur ein, bis der Anwender die (Enter) -Taste drückt. Die eingelesenen Daten speichern wir in der Variablen \$meineVar.

Dabei ist zu beachten, dass das Neue-Zeile-Zeichen mit abgespeichert wird. Da das Neue- Zeile-Zeichen allerdings meist unerwünscht ist (es gehört ja im Grunde nicht zur Eingabe, sondern diene lediglich zum Abschicken der Eingabe), ist es Usus, das Neue-Zeile- Zeichen gleich aus der Eingabe zu entfernen. Dies erledigt die vordefinierte Funktion chomp, der die Variable mit der Eingabe übergeben wird.



*Statt `$meineVar = <STDIN>; chomp($meineVar);` schreiben erfahrene Perl-Programmierer meist verkürzt:  
`chomp($meineVar = <STDIN>);`*

### Listing 17.8: eingabe.pl

```
#!/usr/bin/perl -w
use strict;
my $name;
print("Wie heissen Sie? ");
$name = <STDIN>;
chomp($name);
print("Hallo $name!\n");
```

## Operatoren

Wie jede anständige Programmiersprache verfügt auch Perl über eine Reihe von Operatoren zur Manipulation

elementarer Daten.

## Operatoren für Zahlen

Neben dem Zuweisungsoperator stehen die typischen arithmetischen Operatoren zur Verfügung.

Operator	Bedeutung	Verwendung
op1 + op2	Addition	\$var = 10 + 4; # \$var = 14
op1 - op2	Subtraktion	\$var = 10 - 4; # \$var = 6
op1 * op2	Multiplikation	\$var = 10 * 4; # \$var = 40
op1 / op2	Division	\$var = 10 / 4; # \$var = 2.5

**Tabelle 17.2: Die Grundrechenarten**

Hinzukommen die Operatoren für Exponent und Modulo.

Operator	Bedeutung	Verwendung
op1 ** op2	Exponent	\$var = 10 ** 2; # \$var = 100  Berechnet op1 hoch op2.
op1 % op2	Modulo	\$var = 10 % 3; # \$var = 1  Teilt op1 durch op1 und gibt den dabei verbleibenden Rest als Ergebnis zurück.  op1 und op2 müssen Ganzzahlen sein!

**Tabelle 17.3: Exponent und Modulo**

sowie Inkrement, Dekrement und die kombinierten Zuweisungen.

Operator	Bedeutung	Entspricht
++op1	Inkrement	op1 = op1 + 1
--op1	Dekrement	op1 = op1 - 1
op1 += op2	Addition	op1 = op1 + op2
op1 -= op2	Subtraktion	op1 = op1 - op2
op1 *= op2	Multiplikation	op1 = op1 * op2
op1 /= op2	Division	op1 = op1 / op2
op1 %= op2	Modulo	op1 = op1 % op2
op1 **= op2	Exponent	op1 = op1 ** op2

**Tabelle 17.4: Die kombinierten Zuweisungsoperatoren**

Haben wir irgend etwas vergessen? Ach ja, die Operatoren zum Vergleichen von Zahlen:

Operator	Bedeutung	Verwendung
----------	-----------	------------

==	Gleichheit	<code>\$var1 == \$var2</code>
!=	Ungleichheit	<code>\$var1 != \$var2</code>
<	kleiner	<code>\$var1 &lt; \$var2</code>
>	größer	<code>\$var1 &gt; \$var2</code>
<=	kleiner gleich	<code>\$var1 &lt;= \$var2</code>
>=	größer gleich	<code>\$var1 &gt;= \$var2</code>

**Tabelle 17.5: Die Vergleichsoperatoren für Zahlen**

## Operatoren für Strings

Im Gegensatz zu anderen Programmiersprachen kennt Perl eine Reihe von speziellen Operatoren zur Verarbeitung von Strings.

Neben dem Zuweisungsoperator (=) und dem Umleitungsoperator << (siehe HERE- Texte) erlaubt Perl

- die Aneinanderreihung (Konkatenation) von Strings mit dem .-Operator

```
$str = "Hallo" . $name;
```

- die Wiederholung einzelner Zeichen mit x

```
$rahmen = '*' x 10; # erzeugt *****
```

- und das lexikographische Vergleichen von Strings

Operator	Bedeutung	Verwendung
eq	Gleichheit	<code>\$str1 eq \$str2</code>
ne	Ungleichheit	<code>\$str1 ne \$str2</code>
lt	kleiner	<code>\$str1 lt \$str2</code>
gt	größer	<code>\$str1 gt \$str2</code>
le	kleiner gleich	<code>\$str1 le \$str2</code>
ge	größer gleich	<code>\$str1 ge \$str2</code>

**Tabelle 17.6: Die Vergleichsoperatoren für Strings**

## Kontrollstrukturen

Wie in fast allen strukturierten Programmiersprachen gibt es auch in Perl if-else- Verzweigung und verschiedene Schleifenkonstruktionen.

### Die if-Bedingung

#### Listing 17.9: if.pl

```
#!/usr/bin/perl -w
use strict;
my $eingabe = 0;
print "\nWie alt sind Sie? ";
chomp ($eingabe = <STDIN>);
if ($eingabe >= 10)
```

```

{
print "Sorry, dieses Programm ist nur für Kinder!";
}
else
{
print "Hallo, freut mich Dich kennenzulernen!";
}

```



Mit Hilfe der logischen Operatoren `!`, `&&`, `||` und `^` kann man Vergleiche miteinander kombinieren.

## Die Schleifen

Auch die `for`-Schleife ist nach dem typischen Muster aufgebaut:

```

Ausgabe der ersten 10 Potenzen von 2
my $i;
for ($i = 0; $i <= 10; ++$i)
{
print "2 hoch $i ist \t", 2**$i, "\n";
}

```

Als `while`-Schleife formuliert, sähe der obige Code wie folgt aus:

```

my $i = 0;
while ($i <= 10)
{
print "2 hoch $i ist \t", 2**$i, "\n";
++$i;
}

```



Perl kennt auch eine Mehrfachverzweigung (`switch`-Konstruktion) und noch weitere Schleifenvarianten, auf die wir hier aber nicht weiter eingehen.

## Arrays

Als Arrays bezeichnet man gemeinhin Datenstrukturen, in denen man mehrere gleichartige und zusammengehörende Daten (beispielsweise Messwerte) abspeichern und verwalten kann. In Perl stellen Arrays einen eigenen Variablentyp dar.

### Arrays definieren

Arrays werden in Perl mit dem Präfix `@` definiert:

```
my @meinArray;
```

Wenn man will, kann man dem Array auch gleich bei der Definition die ersten Werte zuweisen:

```
my @meinArray = (1, 2, 3, 4, 5);
```

oder - einfacher noch - einen Wertebereich zuweisen:

```
my @meinArray = (1..5); # Array mit 5 Elementen, in denen die Werte
 # 1, 2, 3, 4 und 5 gespeichert sind
```

Auf die einzelnen Elemente im Array kann man durch Angabe des zugehörigen Index (der standardmäßig mit 0 anfängt) zugreifen:

```
my $var;
$var = $meinArray[0]; # Wert des 1. Elements nach $var kopieren
```

Ooops, da hat sich wohl ein Tippfehler eingeschlichen! Vor meinArray steht statt @ das Präfix \$. Nein, das Präfix \$ ist an dieser Stelle kein Fehler, denn es steht nicht vor einem Array (meinArray), sondern vor einem Element eines Arrays (meinArray[0]) und dieses Element ist ein... Skalar.

## Arrays dynamisch erweitern

Wie in JavaScript sind Arrays in Perl dynamische Strukturen, das heißt, wir können jederzeit neue Elemente in ein Array einfügen oder bestehende Elemente löschen. Perl stellt uns dafür folgende vier Funktionen zur Verfügung.

Funktion	Beschreibung
push	Hängt eines oder mehrere Elemente an das <b>Ende</b> eines Arrays an.  <pre>@array = (1, 2, 3); push(@array, 4);      # (1, 2, 3, 4) @array = (1, 2, 3); push(@array, 4..5);  # (1, 2, 3, 4, 5) @demo = (4, 5); @array = (1, 2, 3); push(@array, @demo); # (1, 2, 3, 4, 5)</pre>
unshift	Hängt eines oder mehrere Elemente an den <b>Anfang</b> eines Arrays an.  <pre>@array = (1, 2, 3); unshift(@array, 4..5); # (4, 5, 1, 2, 3)</pre>
pop	Entfernt das <b>letzte</b> Element eines Arrays und liefert es zurück.  <pre>@array = (1, 2, 3); \$elem = pop @array;  # (1, 2) print \$elem;        # Ausgabe: 3</pre>
shift	Entfernt das <b>erste</b> Element eines Arrays und liefert es zurück.  <pre>@array = (1, 2, 3); \$elem = shift @array; # (2, 3) print \$elem;        # Ausgabe: 1</pre>

Tabelle 17.7: Array-Elemente hinzufügen oder löschen

## Arrays durchlaufen

Mit Hilfe des indizierten Zugriffs, einer for-Schleife und der Anzahl der Elemente im Array ist es kein Problem, ein

Array Element für Element zu durchlaufen und zu bearbeiten.

### Listing 17.10: Auszug aus arrays.pl

```
my @meinArray = (0..20);
my $i;
for($i = 0; $i < @meinArray; ++$i) {
 print $meinArray[$i], " ";
}
```

Wenn Sie eine Array-Variable in einem Umfeld verwenden, wo Perl einen Skalar erwartet (wie zum Beispiel in dem Vergleich `$i < @meinArray`) ersetzt der Perl-Interpreter die Array- Variable durch die Anzahl der Elemente im Array.

Einfacher und sicherer durchläuft man Arrays aber mit der `foreach`-Schleife.

```
my $elem;
foreach $elem (@meinArray) {
 print $elem, " ";
}
```

## Hashes

Ein Hash ist eine weitere Datenstruktur, die insbesondere in Kombination mit Arrays und Referenzen sehr effizient zur Verwaltung und Verarbeitung großer, strukturierter Datenmengen eingesetzt werden kann. Entsprechende Implementierungen werden aber auch schnell recht kompliziert, weswegen wir hier nicht näher darauf eingehen werden. Trotzdem sollte man zumindest wissen, wie Hashes definiert werden und wie man auf in Hashes abgelegte Elemente zugreifen kann.

Stellen Sie sich ein Hash einfach als ein Array vor, auf dessen Elemente man nicht über einen Index in eckigen Klammern, sondern über einen Namen in geschweiften Klammern zugreift.

Wenn man ein neues Hash anlegt, gibt man die einzelnen Elemente als Name/Wert-Paare an - also immer zuerst den Namen, den man später als Index zum Zugreifen auf das Element verwendet, und dann den eigentlichen Inhalt des Elements:

```
%adresse = ("Vorname" => "Sven",
 "Nachname" => "Olsen",
 "Straße" => "Fliederstrasse",
 "Hausnummer" => 23,
 "PLZ" => 81830,
 "Stadt" => "München");
```

Wie Sie sehen, haben auch Hash-Variablen ihr eigenes Präfix: %.

Wenn Sie auf den Wert eines Elements in einem Hash zugreifen wollen, geben Sie den Namen des Elements als Index in geschweiften Klammern an:

```
$person{Vorname} = "Ole"; # Wert ändern/setzen
print $person{Vorname}; # Wert abfragen
```

## Funktionen

Größere Programme haben die Eigenschaft, dass ihr Code schnell sehr unübersichtlich wird. Man kann dem entgegenwirken, indem man Teilprobleme identifiziert und diese in Form von Funktionen löst. Funktionen haben überdies den Vorteil, dass man sie nach erfolgter Definition an beliebiger Stelle aufrufen kann (siehe Kapitel 9.4).

### Funktionen definieren

In Perl werden Funktionsdefinitionen mit dem Schlüsselwort `sub` eingeleitet. Dann folgt der Funktionsname und in geschweiften Klammern der Anweisungsblock zu der Funktion.

```
sub funktionsname {
 # Anweisungen
}
```

## Parameter und Rückgabewert

Perl hat einen recht eigenwilligen Mechanismus zur Übergabe von Argumenten an Funktionen. Es legt alle Argumente in einem Array namens `@_` ab. Dieses Array müssen Sie nicht explizit definieren, es ist automatisch in jedem Perl-Programm verfügbar. In der Funktion können Sie dann die einzelnen Argumente über ihren Index `$_[0]`, `$_[1]` zugreifen.



*Beachten Sie, dass Sie über die Elemente `$_[n]` direkt auf die Variablen zugreifen, die beim Aufruf als Argumente übergeben wurden (die Funktion kann die Variablen aus dem Aufruf ändern). Wenn Sie dies verhindern wollen, weisen Sie die Werte aus dem Array `@_` an lokale Variablen der Funktion zu.*

```
sub funktionsname {
 my param1 = $_[0];
 my param2 = $_[1];
 # weitere Anweisungen
}
```

Aufruf:

```
funktionsname($var1, 312);
```

Soll die Funktion einen Rückgabewert zurückliefern, übergeben Sie diesen zum Schluss der `return`-Anweisung:

```
return $ergebnis;
```

Das folgende Programm nutzt beispielsweise eine Funktion namens `berechne_zins`, um auszurechnen, wie sich ein einmalig eingesetzter Betrag bei fester Verzinsung (mit Zinseszins) vermehrt.

### Listing 17.11: funktionen.pl

```
#!/usr/bin/perl
use strict;
Funktion zur Berechnung des Endkapitals
sub berechne_zins
{
 my $kp = $_[0]; # Startkapital
 my $zs = $_[1]; # Zinssatz
 my $lz = $_[2]; # Laufzeit
 return $kp * (1 + $zs/100.0) ** $lz;
}
my $kapital;
my $zsatz;
Startkapital und Zinssatz von Anwender abfragen
print "\n Programm zur Zinsberechnung\n\n";
print " Wie hoch ist Ihr Startkapital: ";
chomp($kapital = <>);
print " Wie viel Zinsen bekommen Sie (in Prozent): ";
```

```

chomp($zsatz = <>);
Kapitalentwicklung für die nächsten 20 Jahre berechnen
my $laufzeit;
my $endkapital;
for ($laufzeit = 1; $laufzeit <= 20; ++$laufzeit)
{
 $endkapital = berechne_zins($kapital, $zsatz, $laufzeit);
 printf("Endkapital nach %d Jahren: %.2f DM\n", $laufzeit,
 $endkapital);
}

```

## Dateien

Mindestens ebenso wichtig wie die Ein- und Ausgabe über die Konsole ist das Verarbeiten von Daten aus Dateien. Zur Programmierung mit Dateien gehört, dass man weiß, wie man Dateien öffnet, wie man Daten aus Dateien einlesen kann, wie man Daten in Dateien abspeichert und wie man die Dateien zum Schluss wieder schließt.

Das Schreiben einer Datei erfolgt in drei Schritten:

1. Datei öffnen und mit Datei-Handle verbinden.

Zum Öffnen von Dateien verwendet man die Funktion `open`.

```

my $dateiname = "hallo.html";
open(FILEI, ">> $dateiname");

```

Als Argumente übergibt man `open` einen Namen für den einzurichtenden Datei-Handle und den Namen der zu öffnenden Datei. Die Funktion richtet den gewünschten Datei-Handle ein (Datei-Handles stellen unter Perl einen eigenen Datentyp dar), öffnet die angegebene Datei und verbindet diese mit dem Datei-Handle. In unserem Skript wird die Datei jetzt durch den Datei-Handle repräsentiert, alle nachfolgenden Zugriffe auf die Datei erfolgen über den Datei-Handle.



*Wenn die zu öffnende Datei nicht im gleichen Verzeichnis wie das Perl-Skript steht, müssen Sie im Dateinamen auch den Pfad zur Datei angeben. Windows-Anwender, die es gewohnt sind, Verzeichnisse mit einem Backslash \ zu trennen, müssen beachten, dass der Backslash in doppelten Anführungszeichen ein Sonderzeichen ist. Um einen Backslash in einen Dateinamen einzubauen, muss man daher zwei Backslashes hintereinander schreiben: "C:\\datei.txt" (oder einfache Anführungszeichen verwenden). Die meisten Windows-Versionen erlauben aber auch die Verwendung eines einfachen Slashes, "C:/datei.txt", wie es auch unter UNIX/Linux üblich ist.*

Soweit ist alles ganz einfach. Etwas störend ist nur die Zeichenfolge `>>` vor dem Dateinamen (bzw. der Variablen, die den Dateinamen enthält). Diese Zeichenfolge gibt an, zu welchem Zweck die Datei geöffnet werden soll.

Öffnen-Modus	Beschreibung
<code>open(FH, "datei");</code>	Öffnet die Datei zum Lesen.
<code>open(FH, "&lt; datei");</code>	Ist die Datei nicht vorhanden, scheitert die Funktion.
<code>open(FH, "&gt; datei");</code>	Öffnet die Datei zum Schreiben.  Ist die Datei nicht vorhanden, wird sie neu angelegt. Ist die Datei bereits vorhanden, wird ihr alter Inhalt gelöscht.

<code>open(FH, "&gt;&gt; datei");</code>	Öffnet die Datei zum Anhängen.  Ist die Datei nicht vorhanden, wird sie neu angelegt. Ist die Datei bereits vorhanden, wird der neue Inhalt an den alten Inhalt angehängt.
<code>open(FH, "+&lt; datei");</code>	Öffnet die Datei zum Lesen und Schreiben.  (Erfordert meist fortgeschrittene Funktionen zur Dateibehandlung).

**Tabelle 17.8: Öffnen-Modi**

### Fehlerbehandlung für Dateien

Zum Schluss müssen wir noch klären, was geschehen soll, wenn die `open`-Funktion scheitert und die Datei nicht mit dem Datei-Handle verbunden wird. Wenn das Programm ohne die Datei nicht sinnvoll arbeiten kann, empfiehlt es sich den `open`-Aufruf an den Anfang des Skripts zu stellen und mit einem `die`-Aufruf zu kombinieren:

```
open(DATEI, ">> $dateiname")
or
die "\nDatei $dateiname konnte nicht geoeffnet werden\n";
```

Diese Konstruktion sieht man in Perl-Skripten, die mit Dateien arbeiten, sehr häufig. Sie funktioniert wie folgt.

– Wenn `open` die Datei öffnen konnte, liefert sie einen Wert zurück, der dem Wahrheitswert »wahr« entspricht. Da eine OR-Konstruktion bereits »wahr« ist, wenn nur einer der Teilausdrücke »wahr« ist, wird der nachfolgende Ausdruck mit `die` überhaupt nicht mehr ausgewertet (die wird nicht aufgerufen) und das Skript wird normal fortgesetzt.

– Wenn `open` scheitert, liefert `open` den Wahrheitswert »false«, so dass der Interpret auch noch den zweiten Ausdruck der OR-Verknüpfung auswertet. Er ruft also die Funktion `die` auf. Die Funktion `die` gibt noch den als Argument übergebenen String aus und beendet dann sofort das Programm.

#### 2. Daten in Datei schreiben.

Für die Ausgabe von Daten in Dateien verwendet man die gleichen Funktionen wie für die Ausgabe von Daten auf die Konsole: `print` oder `printf`. Der einzige Unterschied ist, dass man den Funktionen als erstes Argument den Datei-Handle übergibt.

```
print DATEI $var1, $var2;
```



*Wenn Sie die Argumente zu einer Dateifunktionen nicht in Klammern erfassen, wird der Datei-Handle nicht durch Komma von den anderen Argumenten getrennt!*

#### 3. Zum Schluss wird die Datei geschlossen.

```
close(DATEI);
```

Das Schließen der Datei führt dazu, dass eventuell noch gepufferte Daten in die Datei geschrieben werden und die Datei mit dem Dateiendezeichen abgeschlossen wird. Wenn Sie das Programm mit geöffnetem Datei-Handle beenden, kann es passieren, dass Daten verloren gehen oder die Datei beschädigt wird.

### Listing 17.12: schreiben.pl

```
#!/usr/bin/perl -w
```

```

use strict;
my $dateiname = "hallo.html"; # Name der zu öffnenden Datei
Datei öffnen und mit Datei-Handle DATEI verbinden
open(DATEI, "> $dateiname")
 or
 die "\nDatei $dateiname konnte nicht geöffnet werden\n";
Anwendername von Konsole einlesen
my $name = "";
print "\n";
print "Geben Sie Ihren Vornamen ein: ";
chomp($name = <STDIN>);
Daten in Datei schreiben
my $htmlcode = <<HERE_HTML;
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Hallo</title>
</head>
<body>
 <h1>Hallo $name !</h1>
</body>
</html>
HERE_HTML
print DATEI $htmlcode;
Datei schliessen
close(DATEI);

```

## Lesen einer Datei

Das Lesen einer Datei erfolgt in drei Schritten:

1. Datei öffnen und mit Datei-Handle verbinden.

Zum Öffnen von Dateien verwendet man wie beim Schreiben die Funktion `open`. Zu beachten ist nur, dass man zum Öffnen im Lese-Modus das `<`-Zeichen vor den Dateinamen stellt (oder nur den Dateinamen angibt, siehe Tabelle 17.8).

```

my $dateiname = "hallo.html";
open(DATEI, "< $dateiname");

```

2. Daten aus Datei lesen.

Wie man die Daten einliest, hängt davon ab, wie der Inhalt der Datei aufgebaut ist. Eine Möglichkeit ist, den Inhalt mit Hilfe des `<>`-Operators und einer `while`-Schleife Zeile für Zeile einzulesen und die einzelnen Zeilen in einer Schleifenvariable zu speichern.

```

while($zeile = <HANDLE>)
{
 # Zeile weiter verarbeiten
}

```

3. Zum Schluss wird die Datei geschlossen.

```

close(DATEI);

```

### Listing 17.13: lesen.pl

```

#!/usr/bin/perl -w

```

```

use strict;
my $dateiname = "hallo.html"; # Name der zu öffnenden Datei
Datei öffnen und mit Datei-Handle DATEI verbinden
open(DATEI, "<$dateiname")
 or
 die "\nDatei $dateiname konnte nicht geöffnet werden\n";
Daten aus Datei einlesen
while(my $zeile = <DATEI>)
{
 print $zeile;
}
Datei schließen
close(DATEI);

```

## Auf Module zurückgreifen

Zu Perl gehört eine umfangreiche Sammlung von Modulen, die jeder Perl-Programmierer in seinen Programmen frei verwenden kann - das sogenannte CPAN. Jedes der CPAN- Module ist einem bestimmten Themengebiet gewidmet und enthält Funktionen (manchmal auch Objekte) zur Lösung typischer Aufgaben aus dem entsprechenden Themengebiet. So gibt es beispielsweise Module mit zusätzlichen mathematischen Funktionen (Math), Module zur Programmierung von Windows-Anwendungen mit Perl (Tk), Grafikbibliotheken (GD, Chart) oder auch zur Erstellung von CGI-Programmen (CGI).

## Module einbinden

Um Elemente eines bestimmten Perl-Moduls in einem Skript verwenden zu können, müssen Sie das Modul mit Hilfe des use-Befehls einbinden:

```

use Tk;

oder

use Text::Wrap;

```

falls sich das Modul in einem Unterverzeichnis befindet.

Wie Sie danach auf die Elemente des Moduls zugreifen können, hängt davon ab, wie das Modul implementiert ist und wie es seine Elemente exportiert.

- Manchmal kann man die Elemente direkt aufrufen, ganz so als wären Sie im eigenen Programm definiert.

```

modulfunktion();

```

- Manchmal muss man dem Elementnamen den Modulnamen voranstellen.

```

Modul:: modulfunktion();

```

- Manchmal, wie im Falle des CGI-Moduls kann man die Elemente explizit in das eigene Programm importieren und sie danach wie selbst definierte Elemente verwenden.

```

use CGI qw(:standard); # Modulelemente importieren
modulfunktion();

```

Was aber wenn man feststellt, dass das betreffende Modul gar nicht installiert ist?

## Module herunterladen und installieren

Grundsätzlich gehört zu jeder Perl-Installation eine mehr oder weniger umfangreiche Auswahl an mitinstallierten

CPAN-Modulen. Dann brauchen Sie die Module, die Sie nutzen wollen, nur einzubinden und die darin enthaltene Funktionalität zu nutzen.

Wenn Sie jedoch versuchen, ein Modul zu verwenden, das auf Ihrem System nicht installiert ist, erhalten Sie nur Fehlermeldungen. Dann ist es an der Zeit, sich ins Internet unter <http://www.perl.com> einzuloggen und sich im CPAN nach dem betreffenden Modul umzuschauen. Einen guten Überblick über das CPAN und die darin enthaltenen Module erhalten Sie unter <http://www.perl.com/CPAN-local/README.html>. Auf dieser Webseite finden Sie auch Informationen über die Installation der CPAN-Module.

Wenn Sie mit einer ActiveState-Version arbeiten (egal ob unter Windows oder Linux), können Sie neue CPAN-Module auch mit Hilfe des Programms **ppm** nachinstallieren.

1. Stellen Sie über Modem oder ISDN-Karte eine Verbindung zum Internet her.
2. Rufen Sie **ppm** über die Konsole auf (unter Windows die MSDOS- Eingabeaufforderung). Es erscheint ein eigener ppm-Prompt.
3. Lassen Sie sich eine Liste der verfügbaren Module anzeigen. (Eventuell sollten Sie zuvor einstellen, nach wie vielen Zeilen die Ausgabe der Liste unterbrochen werden soll.

```
ppm> set more 20
ppm> search
```

4. Wenn Sie das gewünschte Modul (ActiveState spricht von Packages) gefunden haben, lassen Sie es installieren.

```
ppm> install PackageName
```

5. Beenden Sie **ppm** nach erfolgreicher Dateiübertragung.

```
ppm > quit
```



Weitere Informationen zu **ppm** finden Sie in der Online-Dokumentation auf der ActiveState-Website ([www.activestate.com](http://www.activestate.com)).

## Was wir ausgelassen haben

Perl ist einerseits recht einfach zu erlernen, andererseits von überwältigender Komplexität. Dies liegt nicht nur an den vielen zu Perl angebotenen CPAN-Modulen, sondern auch

- an den vielen Symbolen (\$, @, %, qw, @\_ etc)
- den vielen alternativen Syntaxformen (um nicht unnötig Verwirrung zu stiften, haben wir in obigem Programmierkurs weitgehend auf die Beschreibung alternativer Syntaxformen verzichtet)
- verschiedenen, für Programmiersprachen ungewöhnliche Konzepte: Kontextsensitivität, extensiver Gebrauch von Standardvariablen (siehe beispielsweise @\_ für die Übergabe von Argumenten an Funktionen), etc., denen wir in obigem Programmierkurs weitgehend aus dem Weg gegangen sind.

Sie sehen: es gäbe noch viel über Perl zu lernen. Wichtige Themen, die wir nicht angesprochen haben, weil Sie für die nachfolgenden CGI-Beispiele nicht relevant sind, wären beispielsweise:

- Referenzen
- Reguläre Ausdrücke (was wirklich schade ist, denn mit Hilfe der regulären Ausdrücke kann man in Perl-CGI-Programmen vom Browser geschickte Formulareingaben sehr effizient auf verdächtige Zeichen - wie sie beispielsweise in Server Side Includes vorkommen, siehe Kapitel 14.2 und 11.3 - untersuchen).
- Standardvariablen

- Objektorientierte Programmierung in Perl

Auch haben wir nur andeuten können, warum Perl eigentlich Perl heißt. Perl ist nämlich ein Akronym für »Practical Extraction and Reporting Language«, das heißt, die Sprache empfiehlt sich als besonders geeignet zur Verarbeitung von Texten und zur Erstellung von Berichten. Wer Perl in diesem Gebiet einsetzt (das sich auch mit den Aufgaben von CGI- Programmen überschneidet), profitiert unter anderem von

- vielen mächtigen String-Funktionen
- einfacher Umwandlung von Arrays in Strings und umgekehrt
- einer Vielzahl bequemer und variabler Einleseroutinen
- HERE-Texten
- Suchen und Ersetzen mit regulären Ausdrücken

## 17.3 Webseiten dynamisch erzeugen

Nachdem wir uns im vorangehenden Abschnitt ein wenig mit der Perl-Syntax vertraut gemacht haben, wollen wir nun den nächsten Schritt wagen und mit der CGI- Programmierung mit Perl beginnen.

Aus Sicht eines Programms besteht die CGI-Spezifikation aus zwei wichtigen Teilen:

- der eine Teil regelt, wie der Server Daten vom Browser an das Programm weiterleitet
- der andere Teil regelt, wie ein Programm Daten über den Server an den Browser schicken kann.

Wir beginnen mit dem zweiten Teil: dem Zurücksenden von Daten an den Browser

### Daten via CGI zurücksenden

CGI-Programme können jedwede Art von Daten zurückzuliefern, die von den Browsern verarbeitet werden kann: HTML-Dokumente, Grafiken, Sounddateien, etc. Wichtig ist,

- dass das Programm die Daten auf die Konsole (STDOUT) ausgibt
- dass die Daten das richtige Format und einen passenden HTTP-Header haben.

Die erste Bedingung ist nicht schwer zu erfüllen, besagt sie schließlich nichts anderes, als dass wir die Daten mit print oder printf ausgeben sollen - ganz so als würden wir auf die Konsole schreiben (nur dass in diesem Fall der Server die Daten entgegen nimmt und an den Browser leitet).

Auch das richtige Format ist leicht gefunden. Solange wir uns darauf beschränken, Webseiten als Ergebnis der CGI-Programme zurückzuliefern, ist das richtige Format einfach der HTML-Code der Webseite.

Neu für uns ist, dass wir den HTTP-Header mitliefern müssen. Diese Aufgabe übernimmt nämlich ansonsten der Webserver. Jetzt müssen wir uns selbst darum kümmern.

#### HTTP-Header

Die Übertragung von Webseiten über das Internet (oder ein lokales Netz) erfolgt nach den Vorgaben des HTTP-Protokolls (HTTP steht für Hyper Text Transfer Protocol). Dieses Protokoll sieht vor, dass sämtliche übertragenen Dateien von speziellen (teils erforderlichen, teils optionalen) Headern begleitet werden. Diese Header kann man in vier Kategorien aufteilen:

- Allgemeine Header: enthalten unspezifische Daten (beispielsweise eine Datumsangabe)
- Anforderungs-Header: werden vom Browser an den Server geschickt (beispielsweise der Header User-Agent, über den sich der Browser beim Server ausweisen kann)
- Antwort-Header: werden vom Server an den Browser geschickt
- Entity-Header: beziehen sich auf die übertragenen Daten

Verschiedene HTTP-Header sind auch für die CGI-Programmierung interessant. Erwähnen möchten wir hier aber nur zwei: den Location-Header und den Content-type- Header. Einen dieser beiden Header muss das CGI-Programm

zurückliefern.

Location: <http://www.andererserver.com>

Schicken Sie diesen Header, wenn Sie keine Webseite zurückliefern, sondern den Browser zu einer anderen Webseite umleiten wollen (geben Sie einen absoluten URL oder einen URL relativ zu Ihrer Website an).

Content-type: text/html

Schicken Sie diesen Header, um den Datentyp der nachfolgend ausgegebenen und zu übertragenden Daten anzugeben. Für HTML-Seiten lautet der Datentyp text/html, für GIF-Bilder image/gif, für Textdateien text/plain, etc.



*Auch wenn Sie selbst keine weiteren Header angeben, wird der Server vermutlich noch den einen oder anderen Header anhängen (beispielsweise den Server-Header, durch den sich der Server beim Browser ausweisen kann). Beachten Sie aber, dass Sie auf jeden Fall einen der oben erwähnten Header angeben müssen, da diese bei der CGI-Übertragung nicht vom Server generiert werden können!*

Des Weiteren ist zu beachten, dass jeder Header an sich mit der Zeichenkombination Wagenrücklauf (\r) und Zeilenumbruch (\n) abgeschlossen werden muss. Unter Windows erzeugt der Perl-Interpreter aber für die Zeichenkombination statt eines Wagenrücklaufs einen weiteren Zeilenumbruch. Wenn Sie Perl zur CGI-Programmierung unter Windows benutzen oder Ihre Programme einfach nur portabel halten wollen (ausführbar auf Windows- wie auf UNIX/Linux-Rechnern), dürfen Sie die Header daher nur mit \n abschließen (auch wenn es nicht ganz korrekt ist).

Schließlich müssen Sie zwischen dem letzten Header und dem Beginn des HTML-Codes noch einmal einen weiteren Zeilenumbruch (in Vertretung einer Wagenrücklauf/ Zeilenumbruch-Kombination) ausgeben.

Das klingt alles weit komplizierter als es ist. Letzten Endes brauchen Sie sich nur zu merken, dass ein CGI-Programm ein HTML-Dokument auf zweierlei Weise zurückliefern kann

- entweder der HTML-Code der Webseite ist irgendwo abgespeichert und das CGI- Programm leitet den Browser zu der Webseite um (in diesem Fall würde man das CGI-Programm dazu nutzen, in Abhängigkeit von dem vom Browser übergebenen Daten, siehe auch Abschnitt 17.4, zu unterschiedlichen Webseiten umzuleiten).

```
print "Location: /WebPub/Kap02/auto.html\n\n";
```

- oder Sie erzeugen die Webseite dynamisch im CGI-Programm und geben vor dem HTML-Code noch den Content-type-Header aus:

```
print "Content-type: text/html\n\n";
```

## Beispiel

Schauen wir, inwieweit unser Beispielskript aus Abschnitt 17.1.3 diese Voraussetzungen erfüllt.

### Listing 17.14: testcgi.pl

```
#!/usr/bin/perl -w
print "Content-type: text/html\n\n";
print "<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01//EN\" \n\n";
print " \"http://www.w3.org/TR/html4/strict.dtd\"> \n\n";
print "<html>\n\n";
print "<head>\n\n";
```

```

print "<title>CGI-Testprogramm</title>\n";
print "</head>\n";
print "<body>\n";
print "<p>Hallo vom CGI-Programm!</p>\n";
print "</body></html>\n";

```

Als Erstes wird der HTTP-Header ausgegeben. Wir liefern als einziges Header-Feld Content-type zurück, das wir mit einem doppelten Zeilenumbruch beenden (ein Umbruch für das Header-Feld, ein zweiter Umbruch, um das Ende des Headers anzuzeigen). Danach geben wir den HTML-Code der Webseite aus.

Alles ordnungsgemäß.

Etwas lästig sind die vielen print-Aufrufe und Anführungszeichen. Auch müssen wir darauf achten, dass wir doppelten Anführungszeichen innerhalb der Ausgabestrings das Escape-Zeichen \ voranstellen, damit sie der Perl-Interpreter auch als normale auszugebende Anführungszeichen und nicht als Sonderzeichen zum Abschluss des Strings interpretiert. In Perl können wir uns die Ausgabe vereinfachen, indem wir den HTML-Code der Webseite als HERE-Dokument aufsetzen.

### Listing 17.15: hallocgi.pl

```

#!/usr/bin/perl -w
use strict;
my $webseite = <<HERE_ANTWORT;
Content-type: text/html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>CGI-Testprogramm</title>
</head>
<body>
<p>Hallo vom CGI-Programm!</p>
</body>
</html>
HERE_ANTWORT
print $webseite;

```

Na, ist das nicht übersichtlicher?!

Wir können uns die Ausgabe sogar noch weiter vereinfachen, indem wir das CGI-Modul aus dem CPAN verwenden. In obigen Beispiel kann man sich nämlich leicht mit der Ausgabe des Headers vertun. Diesem darf keine Leerzeile vorausgehen, es muss sich aber eine Leerzeile anschließen. Will man auf Nummer sicher gehen, ruft man zur Ausgabe des Headers die Perl-Funktion header auf.

### Listing 17.16: hallocgi2.pl

```

#!/usr/bin/perl -w
use strict;
use CGI qw(:standard);
my $webseite = <<HERE_ANTWORT;
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>CGI-Testprogramm</title>
</head>
<body>
<p>Hallo vom CGI-Programm!</p>
</body>

```

```
</html>
```

```
HERE_ANTWORT
```

```
print header, $webseite;
```

Zu Beginn des Programms wird das CGI-Modul eingebunden und die Standardfunktionen aus dem Modul werden importiert, so dass wir sie ohne Voranstellung des Modulnamens aufrufen können:

```
use CGI qw(:standard);
```

Im HERE-Dokument steht jetzt nur noch der reine HTML-Code, dafür lassen wir vor dem HERE-Dokument (das in \$webseite abgespeichert ist), noch den Rückgabewert der Perl- Funktion header ausgeben:

```
print header, $webseite;
```



*Falls das CGI-Modul auf Ihrem System nicht installiert sein sollte, erhalten Sie vom Perl-Interpreter eine entsprechende Fehlermeldung (testen Sie das Skript dazu auf der Konsole). Sie müssen das Modul dann nachinstallieren (siehe Abschnitt 17.2.10).*

Um das Programm zu testen, gehen Sie wie folgt vor:

1. Führen Sie das Skript zur Probe einmal auf der Konsole aus, um sich zu vergewissern, dass es syntaktisch korrekt ist.
2. Speichern Sie das Skript im **cgi-bin**-Verzeichnis Ihres Servers (siehe 17.1.2).
3. Rufen Sie das Skript direkt vom Browser aus auf, indem Sie den URL des Skripts im Adressfeld eingeben oder setzen Sie eine Webseite auf, die das Skript über einen Hyperlink aufruft.

### Listing 17.17: hallocgi2.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Hallo CGI</title>
</head>
<body>
<p>Wenn Sie Ihren Webserver für die Verarbeitung von CGI-Perl-Skripten
konfiguriert, den Code des <a href="http://localhost/cgi-
bin/hallocgi2.pl">hallocgi-Programms
korrekt abgetippt, das Programm im cgi-bin-Verzeichnis Ihres Servers
abgespeichert und gegebenenfalls noch den Hyperlink in diesem Absatz angepasst
haben, sollte beim Klick auf obigen Hyperlink die vom CGI-Programm dynamisch
erzeugte Webseite angezeigt werden.</p>
</body>
</html>
```

## 17.4 Serverseitige ImageMaps

Statt nur den URL des CGI-Programms an den Server zu schicken, kann der Browser zusätzlich auch noch Daten als Eingaben für das CGI-Programm übertragen. Das CGI- Programm kann dann diese Eingaben auswerten (vorausgesetzt, es wurde entsprechend programmiert), und differenziert auf die Eingaben reagieren. Dies nutzt man beispielsweise bei der Implementierung serverseitiger ImageMaps.

Wie man clientseitige ImageMaps implementiert, haben Sie bereits in Kapitel 2.6 gelernt (lang ist's her). Heute wollen

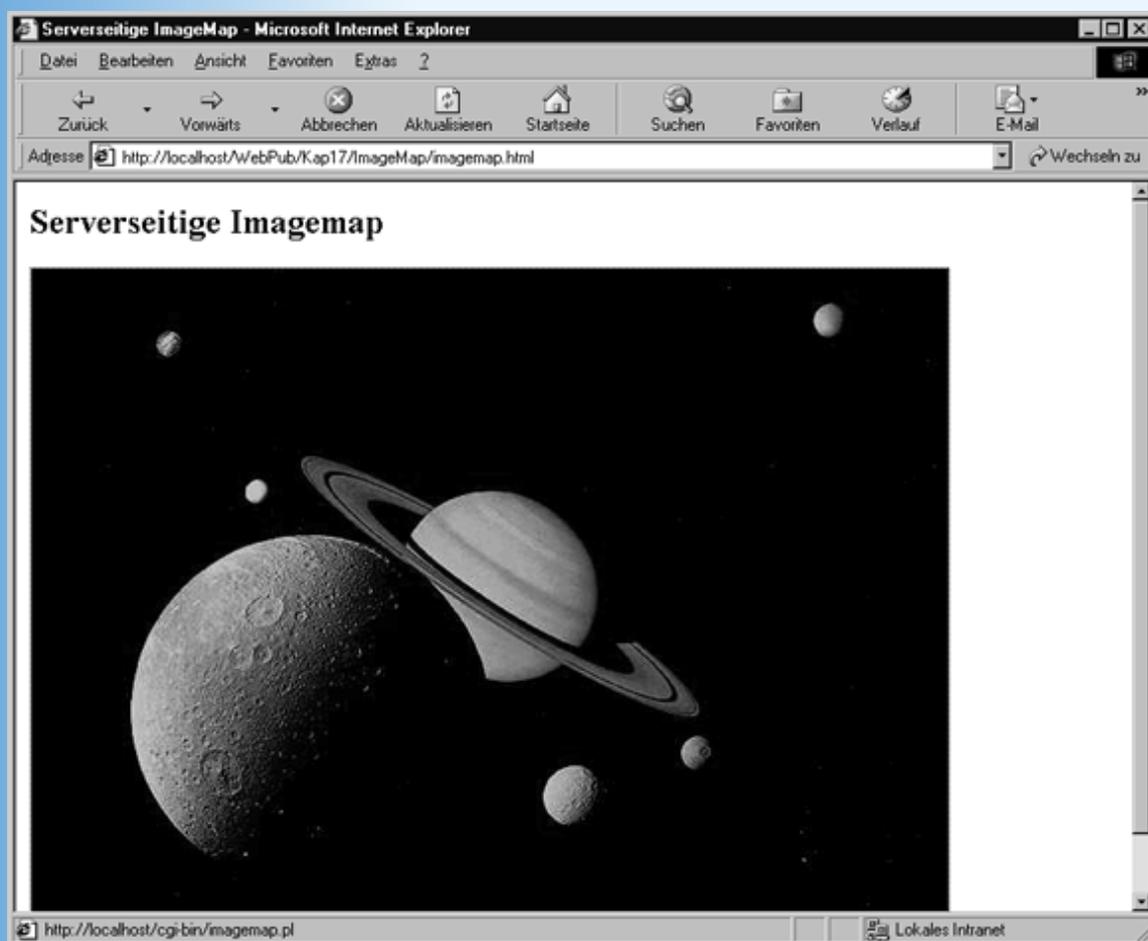
wir Ihnen zeigen, wie man die dort vorgestellte clientseitige ImageMap in eine serverseitige ImageMap verwandeln kann. In der Praxis würde man die ImageMap aus diesem Beispiel aus verschiedenen Gründen fast immer clientseitig realisieren (warum wird bald klar), aber didaktisch macht es Sinn, die beiden Technologien einmal im direkten Vergleich vorzustellen.

## Der HTML-Code

Die clientseitige ImageMap bestand aus dem eigentlichen Bild und einem MAP- Abschnitt, in dem die Flächen definiert waren, die mit Hyperlinks verbunden sein sollten. Die Verbindung zwischen dem Bild und dem MAP- Abschnitt wurde über das usemap- Attribut und den Namen des <map>-Elements hergestellt.

```

<map name="meineMap">
 <area href="s1.html" shape="circle"
 coords="193, 312, 125">
 <area href="s2.html" shape="polygon"
 coords="260, 202, 280, 173, 324, 158, 369, 176, 394, 226, 375,
 274, 346, 290, 321, 287, 299, 232, 300, 232">
</map>
```



**Abbildung 17.3:** Zur Erinnerung: das Bild zur ImageMap

Um diese clientseitige ImageMap in eine serverseitige zu verwandeln, bedarf es dreier Schritte:

- Das <img>-Tag muss in einen Hyperlink gekleidet werden. Dem href-Attribut des Anker-Elements wird der URL des CGI-Programms übergeben.
- Das Attribut usemap wird durch das Boolesche Attribut ismap ersetzt.
- Das <map>-Element fällt ganz weg, seine Aufgabe übernimmt ja das CGI-Programm.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
```

```

<html>
<head>
 <title>Serverseitige ImageMap</title>
</head>
<body>
<h2>Serverseitige Imagemap</h2>

</body>
</html>

```

## Die Übertragung der Koordinaten

Damit das CGI-Programm Mausklücke auf das Bild der ImageMap verarbeiten kann, muss es wissen, an welchen Bildkoordinaten der Mausklick erfolgt. Glücklicherweise brauchen wir uns um die Übertragung der Koordinaten nicht selbst zu kümmern, das übernimmt der Browser. Er hängt an den URL des CGI-Programms ein Fragezeichen (das dem Server mitteilt, dass jetzt Eingabedaten für das Programm folgen) und dann die Koordinaten, die durch ein Komma getrennt sind:

```
http://localhost/cgi-bin/imagemap.pl?143,74
```

Der Server speichert die Eingaben hinter dem Fragezeichen in einer Umgebungsvariablen namens QUERY\_STRING und ruft dann das CGI-Programm auf. Dessen Aufgabe ist es nun, die Koordinaten aus der Umgebungsvariablen QUERY\_STRING einzulesen und zu verarbeiten.



*Eine Umgebungsvariable ist ein Variable, die global auf einem Rechner gespeichert wird und auf die alle Programme, die auf diesem Rechner ablaufen, zugreifen können.*

## Das CGI-Programm

Das Perl-Programm zur Unterstützung der ImageMap erstellen wir in zwei Schritten: zuerst konzentrieren wir uns darauf, die Koordinaten entgegen zu nehmen, dann implementieren wir die eigentliche Unterstützung für die ImageMap.

Die erste Version, die sich darauf beschränkt, die Koordinaten aus QUERY\_STRING einzulesen und zur Kontrolle als HTML-Code auszugeben, sieht wie folgt aus:

### Listing 17.18: Erste Version von imagemap.pl

```

#!/usr/bin/perl -w
use CGI qw(:standard);
use strict;
my $daten = $ENV{'QUERY_STRING'};
my @koord = split(",", $daten);
my $antwort = <<HERE_ANTWORT;
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Hallo</title>
</head>
<body>

```

```
<p>Klick bei $koord[0] und $koord[1].</p>
</body>
</html>
```

HERE\_ANTWORT

```
print header, $antwort;
```

Als Erstes kopiert das Programm den Inhalt der Umgebungsvariablen in die Variable \$daten. Perl verfügt über eine vordefinierte Hash-Variable %ENV, in der beim Programmstart alle auf dem Rechner verfügbaren Umgebungsvariablen abgelegt werden.

Mit folgender Syntax kann man den Inhalt der Umgebungsvariablen QUERY\_STRING aus dem %ENV-Hash herausziehen:

```
my $daten = $ENV{'QUERY_STRING'};
```

Angenommen der Webserver hat folgenden URL an den Server geschickt:

```
http://localhost/cgi-bin/imagemap.pl?143,74
```

Dann ist jetzt in der Variablen \$daten der String "143,74" abgespeichert.

Der nächste Schritt ist, aus diesem String die beiden Koordinaten herauszulesen und in echte Zahlen umzuwandeln. In vielen Programmiersprachen stellt dies ein mittleres Problem dar, nicht aber in Perl, der »Practical Extracton and Reporting Language«. Diese stellt uns eine Funktion namens split() zur Verfügung, mit der man einen String in ein Array verwandeln kann. Dazu übergibt man der Funktion als erstes Argument die Zeichenfolge, die die einzelnen Daten voneinander trennt (in unserem Fall also das Komma), und als zweites Argument den String:

```
my @koord = split(",", $daten);
```

Jetzt steht in \$koord[0] der Wert 143 und in \$koord[1] der Wert 74.

Zum Schluss setzen wir die Antwortseite als HERE-Text auf (wobei Variablenamen durch ihren Wert ersetzt werden) und geben sie aus.

## Das endgültige Programm

Für die Verarbeitung der Mausclicks im CGI-Programm gibt es viele Möglichkeiten. Wir werden im Folgenden prüfen, ob der Besucher der Webseite in einen der größeren Planeten geklickt hat und dementsprechend dem Browser eine Location-Header zurückschicken, der ihn zu einer passenden Webseite umleitet.

### Listing 17.19: imagemap.pl

```
#!/usr/bin/perl -w
use CGI qw(:standard);
use strict;
my $daten = $ENV{'QUERY_STRING'};
my @koord = split(",", $daten);
my $url;
if (($koord[0] > 100) && ($koord[0] < 280) &&
 ($koord[1] > 200) && ($koord[1] < 400))
{
 $url = "http://localhost/WebPub/Kap17/imagemap/jupiter.html";
}
elsif (($koord[0] > 280) && ($koord[0] < 370) &&
 ($koord[1] > 160) && ($koord[1] < 290))
{
 $url = "http://localhost/WebPub/Kap17/imagemap/saturn.html";
}
else
```

```
{
$url = "http://localhost/WebPub/Kap17/imagemap/daneben.html";
}
```

```
print "Location: $url\n\n";
```

Die if-Bedingung prüft, ob die x-Koordinate zwischen 100 und 280 und die y-Koordinate zwischen 200 und 400 liegt - dies entspricht der Abfrage, ob der Mausklick in einem Rechteck liegt, dessen obere linke Ecke bei 100,200 und die rechte untere Ecke bei 280,400 liegt (dieses Rechteck deckt ungefähr den vorderen Planeten ab). Lag der Mausklick innerhalb dieses Rechtecks, wird der URL der Jupiter-Webseite in der Variablen \$url abgespeichert.

Die erste elsif-Bedingung prüft analog, ob der Mausklick im Kernbereich des Saturns lag. Wenn ja wird in der Variablen \$url der URL der Saturn-Webseite abgespeichert

Der abschließende else-Teil wird ausgeführt, wenn der Besucher irgendwo ins Niemandsland geklickt hat.

Zum Schluss wird der ermittelte URL als Location-Header verpackt an den Browser zurückgeschickt. Der Browser wertet den Header aus und lädt die angegebene Datei.

## 17.5 Eigene Daten an CGI-Programme schicken

Analog zu der Art und Weise, wie der Browser die Koordinaten für die Mausklicks an den URL des CG-Programms hängt, können Sie auch selbst in Hyperlinks Daten an den URL eines CGI-Programms hängen.

```
http://server/cgi-bin/antwort.pl?Hallo_CGI
```

Beachten Sie, dass zwischen Perl-Programm und den Daten immer ein Fragezeichen stehen muss und dass Leerzeichen und Sonderzeichen nicht erlaubt sind. Leerzeichen können Sie durch Pluszeichen ersetzen.

## 17.6 Formulare

CGI-Programme werden häufig dazu verwendet, Formulareingaben auf Seiten des Servers zu verarbeiten. Um Programme zur Verarbeitung von Formulareingaben schreiben zu können, muss man wissen, wie die Daten vom Browser an den Server und vom Server an das Programm weitergereicht werden. Im Abschnitt zu den serverseitigen ImageMaps haben wir bereits einen ersten Einblick in die CGI-Datenübertragung bekommen. Diesen wollen wir nun vertiefen und auf eine solide theoretische Basis stellen.

## GET und POST

Die CGI-Spezifikation sieht zwei Wege vor, wie ein CGI-Programm Daten von einem Browser entgegen nehmen kann.

Methode	Beschreibung
GET	<p>Bei dieser Methode hängt der Browser die Zeichenkette an den URL des CGI-Programms an, das aufgerufen werden soll, und schickt den URL des Programms mit den angehängten Daten an den Server. Der Server trennt die Zeichenkette mit den Eingaben wieder von dem URL und speichert sie in der Umgebungsvariablen QUERY_STRING ab. Danach ruft der Server das CGI-Programm auf. Dieses muss so programmiert sein, dass es die Eingabe aus der Umgebungsvariablen QUERY_STRING einliest und auswertet.</p> <p>Ein CGI-Perl-Programm kann die Daten aus dem globalen ENV-Hash auslesen:</p> <pre>my \$daten = \$ENV{'QUERY_STRING'};</pre> <p>Dieses Verfahren ist an sich recht unkompliziert, hat aber den Nachteil, dass die Länge der übergebenen Zeichenkette beschnitten wird, wenn der Speicherplatz für die Umgebungsvariable nicht ausreicht</p>

(üblicherweise 1 KByte = 1024 Zeichen).

POST	<p>Bei dieser Methode schickt der Browser die Daten verpackt in die HTTP-Anfrage an den Server. Der Server speichert die Eingabe-Zeichenkette nicht in einer Umgebungsvariablen, sondern übergibt Sie über die Standardeingabe an das CGI-Programm. Lediglich die Länge der codierten Eingabe-Zeichenkette wird in einer Umgebungsvariablen (CONTENT_LENGTH) abgelegt.</p> <p>Ein CGI-Perl-Programm kann die Daten mit Hilfe der read-Funktion einlesen:</p> <pre>\$groesseDaten = \$ENV('CONTENT_LENGTH'); read (STDIN, \$form_info, \$groesseDaten);</pre>
------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Tabelle 17.9: CGI-Methoden zur Datenübergabe**

Auf welchem Weg die Daten übertragen wurden, wird in der Umgebungsvariablen REQUEST\_METHOD abgespeichert.

### Umgebungsvariablen

Eine Umgebungsvariable ist ein Variable, die global auf einem Rechner gespeichert wird und auf die alle Programme, die auf diesem Rechner ablaufen, zugreifen können. Auf Webservern stehen zur CGI-Unterstützung unter anderem folgende Umgebungsvariablen zur Verfügung

- GATEWAY\_INTERFACE - Version des CGI-Protokoll. Standardmäßig CGI/1.1.
- SERVER\_PROTOCOL - Version des HTTP-Protokolls. Meist HTTP/1.0.
- REQUEST\_METHOD - Entweder POST oder GET.
- PATH\_INFO - Daten, die nach einem Slash an einen URL gehängt werden. Wird normalerweise verwendet, um einen Pfad relativ zum Dokument-Wurzelverzeichnis anzugeben.
- PATH\_TRANSLATED - Der vollständige Pfad von PATH\_INFO.
- QUERY\_STRING - Enthält die Eingabedaten, wenn die GET-Methode verwendet wird. Unabhängig von der Methode enthält QUERY\_STRING die Daten, die nach einem Fragezeichen (?) an den URL gehängt werden.
- CONTENT\_TYPE - Beschreibt, wie die Daten kodiert sind. Normalerweise application/x-www-form-urlencoded. Wird für das Hochladen von Dateien via HTTP auf multipart/form-data gesetzt.
- CONTENT\_LENGTH - Länge der Eingabe, wenn Sie die POST-Methode verwenden.
- SERVER\_SOFTWARE - Name und Version der Server-Software.
- SERVER\_NAME - Host-Name der Maschine, auf der der Server läuft.
- SERVER\_ADMIN - E-Mail-Adresse des Web-Server-Verwalters.
- SERVER\_PORT - Port, auf dem der Server läuft - normalerweise 80.
- SCRIPT\_NAME - Name des CGI-Programms.
- DOCUMENT\_ROOT - Dokumentenwurzelverzeichnisses auf dem Server.
- REMOTE\_HOST - Name der Client-Maschine, die Informationen anfordert oder sendet.
- REMOTE\_ADDR - IP-Adresse der mit dem Server verbundenen Client-Maschine.
- HTTP\_ACCEPT - Liste von durch Kommata getrennter MIME-Typen, die der Browser interpretieren kann.
- HTTP\_USER\_AGENT - Name, Version und normalerweise auch Plattform des Browsers.
- HTTP\_REFERER - Speichert den URL der Seite, von der aus auf die aktuelle Seite verwiesen wurde.

### URL-Codierung

Unabhängig davon, welchen Weg der Browser wählt, muss er die Daten für die Übertragung an den Server noch codieren (Leerzeichen werden durch Plus-Zeichen, Sonderzeichen durch einfache Zeichenfolgen, Eingaben aus

Formularfeldern als Name=Wert-Paare codiert).

Angenommen der Besucher einer Webseite tippt in ein Eingabefeld seinen Namen ein:

Ihr Name : Dirk Louis

Wenn das Eingabefeld im HTML-Code die Name-ID Feld1 hat und das Formular so konfiguriert ist, dass es seine Eingaben zur Auswertung per GET an ein CGI-Programm namens *cgiskript.pl* schickt, so würde der fertige URL, der vom Browser an den Server gesendet wird, wie folgt aussehen:

```
http://server/cgi-bin/cgiskript.pl?Feld1=Dirk+Louis
```

## Formulareingaben entgegen nehmen

Wird ein CGI-Programm vom Server zur Verarbeitung von Formulardaten aufgerufen, muss es als erstes feststellen, auf welchem Weg (GET oder POST) ihm die Daten übergeben wurden, dann muss es die Daten einlesen und dekodieren. Jetzt erst kann es darangehen, die Daten zu verarbeiten.

Grundsätzlich kann man all dies mit Hilfe der Umgebungsvariablen REQUEST\_METHOD, QUERY\_STRING und CONTENT\_LENGTH selbst implementieren. Man kann es sich aber auch einfacher machen und das Perl-Modul CGI und dessen Funktionen verwenden. Dann braucht man nämlich nur mit Hilfe des Import-Tags :standard die Standardfunktionalität des Moduls einbinden und kann danach die Eingaben problemlos mit Hilfe der Funktion param einlesen:

```
use CGI qw(:standard);
...
my $name = param('name');
```

Doch eines nach dem anderen! Beginnen wir damit, eine Webseite mit einem passenden Testformular aufzusetzen.

## Das Formular

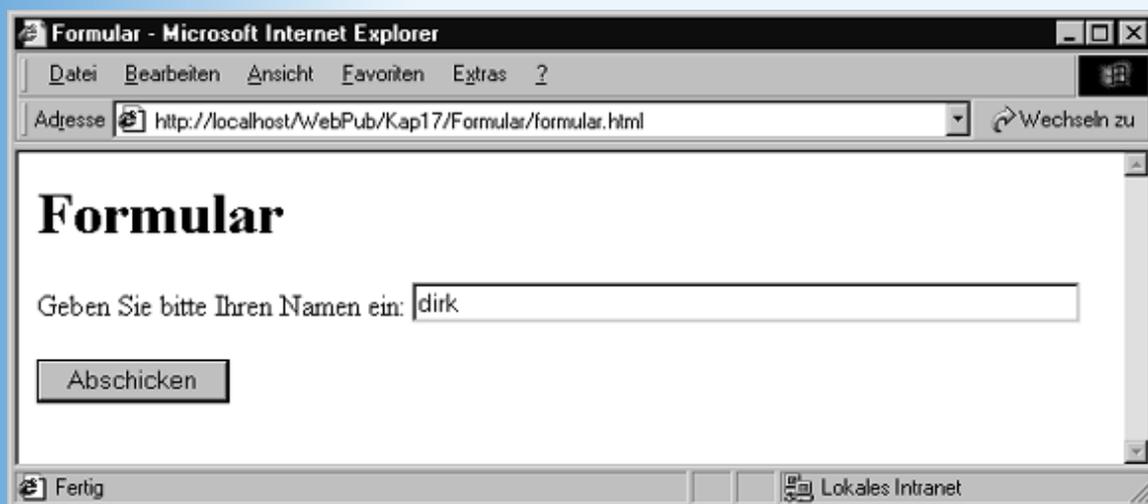


Abbildung 17.4: Das Formular im Browser

Der HTML-Code des Formular aus Abbildung 17.4 sieht wie folgt aus:

### Listing 17.20: formular.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Formular</title>
```

```

</head>
<body>
<h1>Formular</h1>
<form action="http://localhost/cgi-bin/formular.pl" method="get">
 <p>Geben Sie bitte Ihren Namen ein: <input name="name" size="50" />
 </p><p><input type="submit" value="Abschicken" /></p>
</form>
</body>
</html>

```

Zwei Dinge sind aus Sicht des CGI-Programmierers am HTML-Code des Formulars besonders wichtig:

- Jedem Formularfeld kann mit Hilfe des Attributs `name=` ein Name zugewiesen werden. Beim Abschicken der Formulareingaben sendet der Browser die Eingaben zusammen mit dem Namen des dazugehörigen Formularelements (also beispielsweise `name="Peter"`). So kann das Perl-Skript, das die Daten entgegen nimmt, anhand der Namen feststellen, welche Eingaben zu welchen Formularelementen gehören.
- Über das `action`-Attribut des Formulars kann man festlegen, von welchem Skript die Formulareingaben verarbeitet werden sollen und wo dieses Programm zu finden ist (im Listing *formular.html* beispielsweise wird als Bearbeiter das Skript *formular.pl* aus dem *cgi-bin*-Verzeichnis des Webservers angegeben).
- Über das `method`-Attribut des Formulars kann man festlegen, ob die Daten via der GET-Methode (`method="get"`) oder der POST-Methode (`method="post"`) übertragen werden sollen. Wenn Sie kein `method`-Attribut angeben, verwendet der Browser standardmäßig die GET-Methode.

Damit steht fest: das verarbeitende Skript muss *formular.pl* heißen, im Verzeichnis *cgi-bin* stehen und die Eingabe aus dem Feld mit dem Namen »name« verarbeiten.

## Das CGI-Skript

### Listing 17.21: formular.pl

```

#!/usr/bin/perl -w
use CGI qw(:standard);
use strict;
my $name = param('name');
my $antwort = <<HERE_ANTWORT;
 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
 <html>
 <head>
 <title>Hallo</title>
 </head>
 <body>
 <h1>Hallo $name!</h1>
 <p>Es freut mich, Dich auf meiner Homepage willkommen heißen zu dürfen</p>
 </body>
 </html>
HERE_ANTWORT
print header, $antwort;

```

Zuerst wird das CGI-Modul eingebunden. Dieses stellt uns die Methode `param` zur Verfügung, die das Entgegennehmen der Formulareingaben vereinfacht. Wir brauchen der Funktion nur den Namen des Formularfeldes zu übergeben, an dessen Inhalt wir interessiert sind, und die Funktion liefert uns diesen zurück:

```
my $name = param('name');
```

Der Rest des Skripts besteht darin, in HTML-Code eine Antwortseite aufsetzen. Für diese Aufgabe gibt es im CGI-Modul übrigens eine Reihe von vordefinierten Funktionen (`start_html`, `start_form`, `end_form`, `table`, `h1`, `end_html`, etc.). Wir ziehen es aber vor, den HTML-Code der zurückzuliefernden Seite in Form eines HERE-Dokuments aufzusetzen, da der typische Aufbau der HTML-Seite dann sichtbar bleibt und im Bedarfsfall leichter zu korrigieren oder

anzupassen ist.

Zum Schluss wird der HTML-Code mit `print` an die Standardausgabe geschickt - nicht jedoch ohne vorher die CGI-Methode `header` aufzurufen, die den HTTP-Header zurückliefert.

Speichern Sie jetzt das Skript unter dem Namen *Formular.pl* im *cgi*-Verzeichnis Ihres Webservers (wenn das *CGI*-Verzeichnis Ihres Servers anders lautet, müssen Sie die Pfadangabe in *Formular.html* ändern). Laden Sie die Datei *Formular.html* in Ihren Browser, geben Sie Ihren Namen in das Eingabefeld des Formulars ein und klicken Sie auf den Abschicken-Schalter. Wenn alles korrekt eingerichtet ist, sollte kurz darauf die Antwortseite im Browser erscheinen.

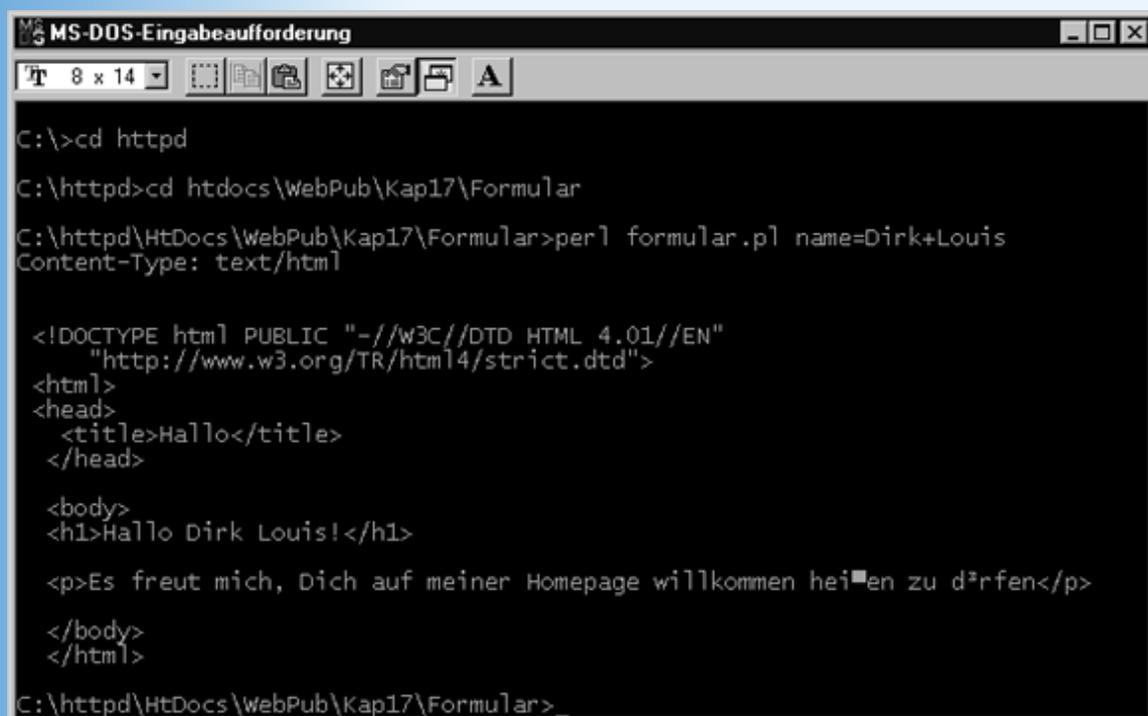


Abbildung 17.5: Antwortseite im Browser

## CGI-Skripte von der Konsole aus testen

Falls Sie keine Antwortseite sehen, könnte dies daran liegen, dass sich beim Aufsetzen des Skripts ein Fehler eingeschlichen hat. Dann empfiehlt es sich, die Korrektheit des Skripts erst einmal von der Konsole aus zu testen. Syntaxfehler können so schnell festgestellt werden. Was macht man aber, wenn der Fehler im logischen Aufbau des Skripts liegt oder die Ausgabe des Skripts fehlerhaft ist.

Um solche Fehler aufspüren zu können, ist es erforderlich, dass das Skript über die Konsole die gleichen Daten entgegen nimmt, die es auch vom Browser erhalten würde. Dazu müssen Sie die Eingaben in der Kommandozeile als `name=wert`-Paare übergeben und Leerzeichen im `wert`-Teil durch `+`-Zeichen ersetzen.



## Abbildung 17.6: CGI-Skripte von der Konsole aus testen



*Das funktioniert allerdings nur, wenn das Skript die Daten nach der POST-Methode entgegen nehmen kann. Die CGI-Methode `param()` unterstützt GET- und POST-Methode.*

Ansonsten schreiben die meisten Webserver Fehler, die beim Aufruf einer Seite oder eines CGI-Programmes auftreten, in eine Error-Datei. (Lesen Sie in der Dokumentation Ihres Webserver nach, wie diese Datei heißt und wo sie abgespeichert ist.)

## 17.7 Verborgene Formularfelder

Wenn der Websurfer den Submit-Schalter eines Formulars drückt, erstellt der Browser die Name/Wert-Paare für alle Steuerelemente im Formular (die Formularfelder) und schickt diese zusammen mit dem URL des verarbeitenden CGI-Programms an den Server. Manchmal reicht uns dies jedoch nicht und wir würden gerne - zusammen mit den Formulareingaben - noch weitere Daten an das CGI-Programm schicken.

Dies könnte beispielsweise der Fall sein, wenn Sie ein CGI-Programm zum Verarbeiten mehrerer Formulare haben (vielleicht führen Sie eine größere Umfrage durch, so dass Sie die Fragen auf mehrere Formulare verteilt haben, die nacheinander von dem CGI- Programm erzeugt und ausgewertet werden). Dann wäre es praktisch, wenn zusammen mit den Formulareingaben auch eine Kennnummer für das Formular gesendet würde, zu dem die aktuellen Daten gehören.

Glücklicherweise ist dies sogar erstaunlich einfach zu realisieren. Man baut einfach verborgene Formularfelder (input-Elemente mit dem Typ `hidden`) in das Formular ein, weist die zu übertragenden Daten deren `value`-Attributen zu und nutzt ansonsten die automatische Datenübertragung durch den Browser.

```
<form action="/cgi-bin/meinprogramm.pl" method="post">
 <input type="hidden" name="formularNr" value="1" />
 <table border="0" cellspacing="0" cellpadding="10">
 <tr>
 <td> Name: <input name="name" size="30" /></td>
 </tr>
 ...
 </table>
</form>

<form action="/cgi-bin/meinprogramm.pl" method="post">
 <input type="hidden" name="formularNr" value="2" />
 <table border="0" cellspacing="0" cellpadding="10">
 <tr>
 <td> Lieblingsfarbe: <input name="farbe" size="30" /></td>
 </tr>
 ...
 </table>
</form>
```

## 17.8 Gästebücher

Was ein Gästebuch ist, wissen Sie sicher. Zahllose private Homepages verfügen über ein Gästebuch, in das sich die Besucher der Website eintragen und das sie sich natürlich auch ansehen können.

Im Folgenden wollen wir uns anschauen, wie man ein solches Gästebuch mit Hilfe von CGI und Perl realisieren kann. Die hier gezeigte Implementierung besteht aus drei Elementen:

- einem Formular, über das sich die Besucher der Website in das Gästebuch eintragen können.
- dem Perl-Skript, das diese Formulareingaben verarbeitet und an das Gästebuch anhängt
- einer HTML-Datei, die das eigentliche Gästebuch darstellt.

## Das Gästebuch

Beginnen wir mit dem Gästebuch.

1. Legen Sie unter dem Dokumentenverzeichnis Ihres Webservers ein neues Unterverzeichnis namens *Gaestebuch* an.
2. Setzen Sie das Grundgerüst des Gästebuchs auf. Achten Sie vor allem auf die letzte Zeile, die genauso wie hier abgedruckt aussehen muss.

### Listing 17.22: gaestebuch.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<title>Gästebuch</title>
</head>
<body>
<h1> Mein Gästebuch </h1>
</body></html>
```

3. Speichern Sie die Datei des Gästebuchs als *gaestebuch.html* im Verzeichnis *Gästebuch* (und achten Sie besonders unter UNIX/Linux darauf, dass das Perl-Skript Schreibrechte hat).

## Das Formular zum Eintragen ins Gästebuch

4. Setzen Sie den HTML-Code für das Formular zum Eintragen in das Gästebuch auf.

Das folgende Formular kennen Sie bereits aus Kapitel 3. Es verfügt über drei Eingabefelder und ein mehrzeiliges Textfeld mit den Namen: name, email, website und kommentar. Als CGI-Programm zur Bearbeitung der Formulardaten wurde */cgi-bin/gaestebuch.pl* angegeben (wenn das CGI-Verzeichnis Ihres Servers anders lautet, müssen Sie die Pfadangabe entsprechend anpassen).

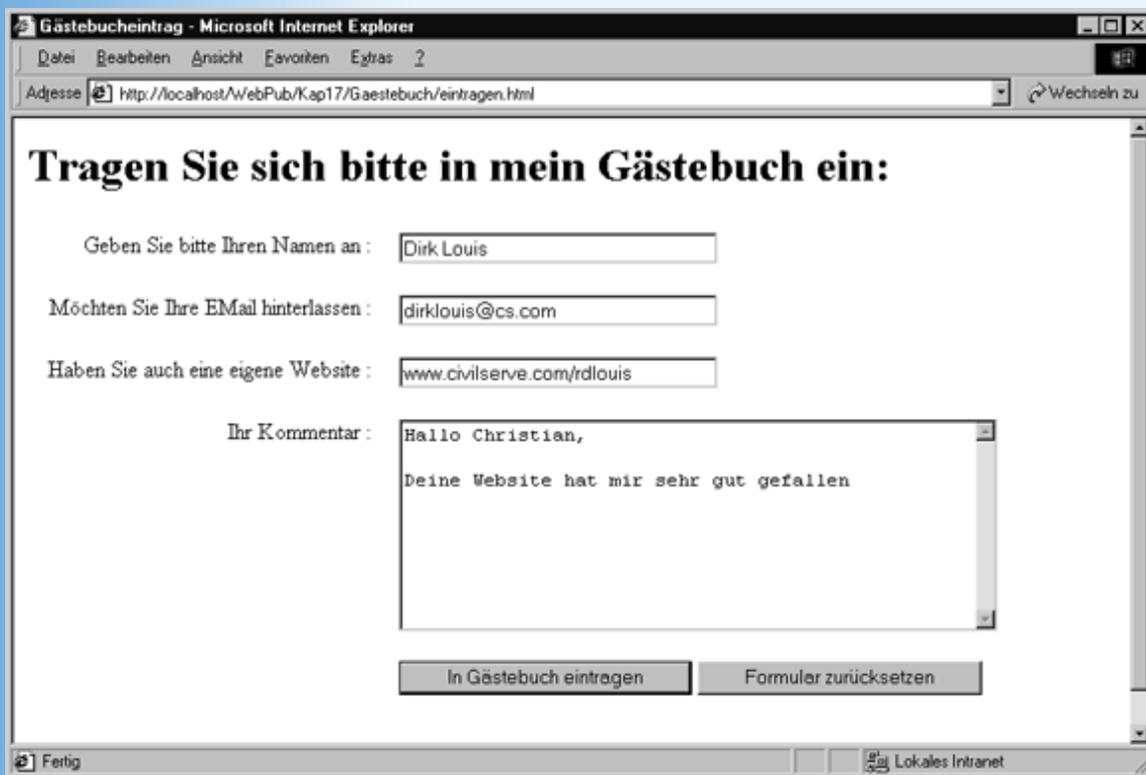
### Listing 17.23: eintragen.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Gästebucheintrag</title>
</head>
<body>
<h1>Tragen Sie sich bitte in mein Gästebuch ein:</h1>
<form action="/cgi-bin/gaestebuch.pl" method="post">
 <table border="0" cellspacing="0" cellpadding="10">
 <colgroup span=2">
 <col width="230">
 <col width="450">
 </colgroup>
 <tr>
 <td align="right" valign="top" width="230">
 Geben Sie bitte Ihren Namen an : </td>
 <td><input name="name" size="30" maxlength="50" /></td>
 </tr>
 <tr>
 <td align="right" valign="top" width="230">
 Möchten Sie Ihre EMail hinterlassen : </td>
 <td><input name="email" size="30" maxlength="50" /></td>
 </tr>
 <tr>
```

```

<td align="right" valign="top" width="230">
 Haben Sie auch eine eigene Website : </td>
<td><input name="website" size="30" maxlength="50" /></td>
</tr>
<tr>
<td align="right" valign="top" width="230">Ihr Kommentar :</td>
<td><textarea name="kommentar" rows="9"
 cols="50"></textarea></td>
</tr>
<tr>
<td> </td>
<td><input type="submit" value="In Gästebuch eintragen" />
 <input type="reset" value="Formular zurücksetzen" /></td>
</tr>
</table>
</form>
<hr />
</body>
</html>

```



**Abbildung 17.7: Das Formular zum Eintragen in das Gästebuch**

5. Speichern Sie die Datei des Eingabeformulars als *eintragen.html* im Verzeichnis *Gästebuch*.

## Das Perl-Programm

6. Setzen Sie das Perl-Programm auf.

### Listing 17.24: gaestebuch.pl

```

#!/usr/bin/perl -w
use CGI qw(:standard);
use strict;
***** Gästebuch öffnen *****
my $gaestebuch =
 "../htDocs/WebPub/Kap17/Gaestebuch/ gaestebuch.html";

```

```

open(BUCH, "+< $gaestebuch")
 or
 die "\nDatei $gaestebuch konnte nicht geoeffnet
 werden\n";
***** Eingaben aus Formular lesen *****
my ($name, $email, $website, $kommentar) =
 (param('name'), param('email'), param('website'),
 param('kommentar'));
***** Eintrag an Gästebuch anhängen *****
my $neuereintrag = <<HERE_EINTRAG;
<p>$name schrieb: </p>
<p><i>$kommentar</i></p>
<p>$name ist über $email erreichbar
 und unterhält eine eigene Website ($website).
</p>
<hr>
HERE_EINTRAG

seek(BUCH, -14, 2);
print BUCH "$neuereintrag\n</body></html>";
close(BUCH);
***** Dankseite zurücksenden *****
my $danke = <<HERE_DANKE;
<html>
<head>
<title>Danke!</title>
</head>
<body>
<h1>Danke!</h1>
<p>Danke, dass Sie sich in mein Gästebuch
 eingetragen haben!</p>
<p>Haben Sie denn auch schon einmal in das
<a href="http://localhost/
 ~/WebPub/Kap17/Gaestebuch/gaestebuch.html">Gästebuch
 hineingeschaut?</p>
<p> </p>
<p><i>So long</i></p>
</body></html>
HERE_DANKE
print header, $danke;

```



*Der Pfad zur Gästebuchdatei muss für Ihren Server angepasst werden!*

Als Erstes öffnet das Skript die Gästebuchdatei:

```

my $gaestebuch =
 "../htDocs/WebPub/Kap17/Gaestebuch/ gaestebuch.html";
open(BUCH, "+< $gaestebuch")
 or
 die "\nDatei $gaestebuch konnte nicht geoeffnet
 werden\n";

```

Gegen den Code an sich ist nichts zu sagen, er könnte jedoch noch ein wenig verbessert werden. Bedenken Sie, dass das Skript unter Umständen von zwei verschiedenen Besuchern Ihrer Website mehr oder weniger gleichzeitig

aufgerufen werden könnte. Wenn die beiden Instanzen dann in die Datei schreiben, kann es passieren, dass Daten verloren gehen oder gar die ganze Datei korrumpiert wird. Sie können dies verhindern, indem Sie die Datei nach dem Öffnen mit Hilfe der Funktion flock aus dem Fcntl-Modul sperren:

```
use Fcntl qw(:flock);
...
flock(BUCH, LOCK_EX);
```

In den darauf folgenden Zeilen werden die Inhalte der Formularfelder ausgelesen:

```
my ($name, $email, $website, $kommentar) =
 (param('name'), param('email'), param('website'),
 param('kommentar'));
```

Hierzu ist anzumerken, dass das Skript nicht prüft, ob für alle Felder sinnvolle Eingaben vorliegen. Wir haben im Beispiel darauf verzichtet, um den Code nicht unnötig zu komplizieren. Wenn Sie CGI-Skripte schreiben, die Eingaben verarbeiten, sollten Sie die Eingaben aber unbedingt prüfen (mit `if(!defined param('name'))`).



*Achten Sie auch darauf, was Sie mit den Formulareingaben machen. Böswillige Hacker könnten statt dem erwarteten Namen oder Kommentar einen Betriebssystembefehl, eine Perl-Befehlsfolge oder ähnlichen Code eingeben, der je nachdem wie die Eingabe weiterverarbeitet wird, ein riesiges Loch in das Sicherheitsnetz Ihres Servers reißen kann.*

Unter Verwendung der Formulareingaben wird - in Form eines HERE-Dokuments - ein neuer Eintrag für das Gästebuch aufgesetzt. Dieser wird anschließend an das Gästebuch angehängt. Allerdings wollen wir nicht, dass der neue Eintrag an die abschließende `</body></html>`-Zeile angehängt wird, sondern er soll diese ersetzen. Wie kann man dies bewerkstelligen? Jeder Datei-Handle verfügt über einen Positionsmarker, der angibt, an welche Stelle der Datei als Nächstes geschrieben bzw. von wo gelesen wird. Diese Positionsmarke kann man mit Hilfe der Perl-Funktion seek verschieben. Dazu übergibt man seek den Datei-Handle, die Anzahl der Zeichen, um die verschoben werden soll, und die Position, ab der verschoben werden soll. Der Aufruf

```
seek(BUCH, -14, 2);
```

verschiebt die Positionsmarke vom Dateende an (Argument 2) um 14 Zeichen nach vorne (Argument -14).

Danach werden der neue Eintrag und eine neue abschließende Zeile in die Datei geschrieben:

```
print BUCH "$neuereintrag\n</body></html>";
```

Zu guter Letzt wird eine Antwortseite zurückgeliefert, die dem Besucher der Webseite anzeigt, dass seine Eingaben verarbeitet wurden, und die ihm einen Link zum Anzeigen des Gästebuchs anbietet.

7. Speichern Sie das Skript unter dem Namen *gaestebuch.pl* im *cgi*-Verzeichnis Ihres Webservers. Wenn Sie unter UNIX/Linux arbeiten, müssen Sie die Zugriffsrechte ändern, damit das Skript vom Webserver ausgeführt werden kann (beispielsweise `chmod 755 gaestebuch.pl`) und auch sicherstellen, dass das Perl-Skript Schreib- und Leserecht für die Gästebuchwebseite besitzt (beispielsweise `chmod 666 gaestebuch.html`).
8. Rufen Sie jetzt das HTML-Dokument mit dem Formular über Ihren Browser auf, füllen Sie die Felder des Formulars aus und klicken Sie auf den Schalter. Wenn alles korrekt eingerichtet ist, sollten kurz darauf die Antwortseite im Browser erscheinen.

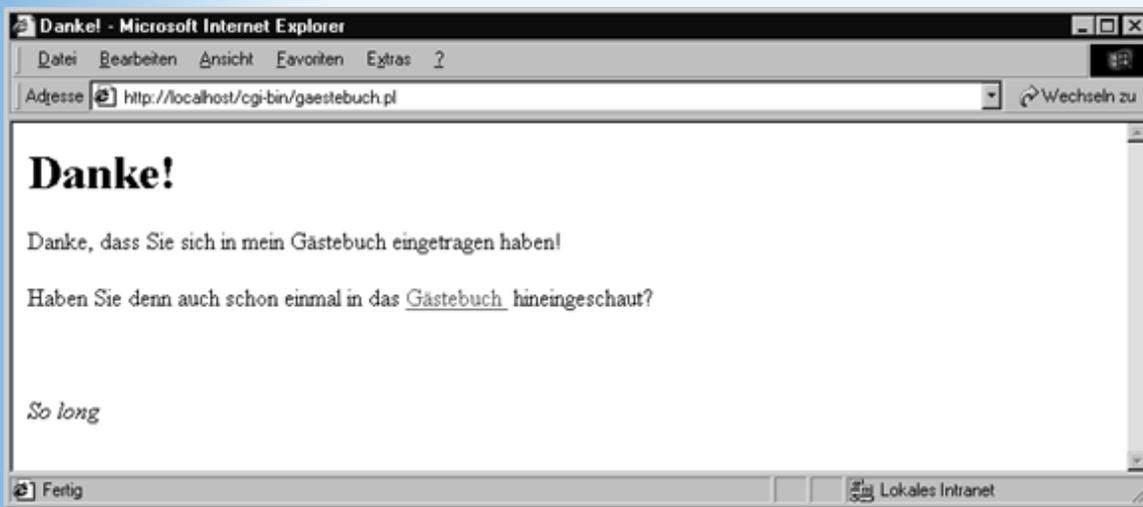


Abbildung 17.8: Von gaestebuch.pl generierte Antwortseite

9. Über den Link können Sie das Gästebuch einsehen.



Abbildung 17.9: Das Gästebuch



Wenn Sie das Gästebuch beim Testen des Skripts wiederholt aufrufen, kann es passieren, dass die zuletzt eingetragenen Formularangaben nicht angezeigt werden. Dies muss nicht am Skript liegen, sondern kann damit zu tun haben, dass Ihr Browser den Inhalt der Gästebuchdatei aus seinem Cache rekonstruiert. Aktualisieren Sie dann die Seite oder löschen Sie den Cache.

## 17.9 Zusammenfassung

Die CGI-Spezifikation legt fest, wie Daten vom Browser an ein Programm auf einem Server übertragen werden können, wie das Programm die Daten entgegen nehmen und wie es seine Ausgabe an den Browser zurücksenden kann.

In diesem Kapitel haben wir uns sowohl mit der Theorie der CGI-Datenübertragung (GET- und POST-Methode, URL-Codierung, CGI-Umgebungsvariablen auf dem Server) als auch mit der Implementierung von CGI-Programmen beschäftigt und dazu eine neue Programmiersprache, Perl, kennen gelernt.

Perl ist eine an sich recht einfach zu erlernende Programmiersprache (kein Vergleich zu Java!) - auch wenn die Vielzahl von Symbolen und alternativen Syntaxformen, die es in der Sprache gibt, den Anfänger eher verwirren. Dass sich die Beschäftigung mit Perl für Webdesigner und Webprogrammierer durchaus lohnen kann, beweist allein schon die große Zahl der weltweit installierten CGI-Perl-Programme. Und sollte es darüber hinaus noch weiterer Argumente bedürfen, so braucht man nur die Beispiele im zweiten Teil dieses Kapitels durchzugehen und sich anzuschauen, mit welcher Eleganz und Leichtigkeit Aufgaben wie das Parsen von QUERY\_STRING (Beispiel zur serverseitigen ImageMap) das Lesen und Schreiben von Dateien (Gästebuch-Beispiel) oder das Ausgeben von HTML-Seiten (Beispiel zur dynamischen Erstellung von Webseiten, Gästebuch) in Perl realisiert werden können.

## 17.10 Fragen und Antworten

### Frage:

**Kann man mit Perl nur CGI-Programme schreiben?**

### Antwort:

*Nein, mit Perl können Sie praktisch jede beliebige Art von Programm schreiben (mit Hilfe des Tk-Moduls sogar Programme mit grafischer Benutzeroberfläche). Dass Perl so häufig in Zusammenhang mit der CGI-Programmierung erwähnt wird, hängt einfach damit zusammen, dass die Stärken von Perl in der Textverarbeitung liegen - was für die CGI-Programmierung sehr vorteilhaft ist.*

### Frage:

**Wie ist es möglich, dass die CGI-Methode param() sowohl via GET als auch via POST verschickte Formulareingaben einlesen und an das CGI-Programm zurückliefern kann?**

### Antwort:

*Nun, ganz einfach. Bei jeder CGI-Datenübertragung wird der gewählte Übertragungsweg (GET oder POST) in der Umgebungsvariablen REQUEST\_METHOD abgelegt. Die param()-Methode fragt den Wert von REQUEST\_METHOD ab und liest die Daten je nach verwendeter Übertragungsmethode aus der Umgebungsvariable QUERY\_STRING (GET-Methode) oder der Standardeingabe (POST-Methode) ein.*

### Frage:

**Gibt es Adressen im Web, von wo man fertige CGI-Programme herunterladen kann?**

### Antwort:

*Die gibt es: <http://www.cgi-resources.com>, <http://www.worldwidemart.com/scripts>, <http://www.freecode.com> oder <http://www.scriptsearch.com>.*

## 17.11 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

1. Mit welchen Präfixen beginnen in Perl skalare Variablen und Array-Variablen?
2. Welcher Unterschied besteht zwischen Perl-Strings in einfachen und doppelten Anführungszeichen?
3. Mit welchem Operator prüft man in Perl, ob der Wert eines Skalars \$var1 kleiner ist als der Wert von Skalar \$var2?
4. Mit welchem Operator prüft man in Perl, ob der String in einem Skalar \$str1 kleiner ist als der String in dem Skalar \$str2?
5. Was macht man, wenn man in Perl eine Lösung für eine Aufgabe sucht, mit der sich vermutlich schon andere Perl-Programmierer herumschlagen mussten?

6. Wozu dient die Perl-CGI-Methode header?
7. Wie kann man zusammen mit den Formulareingaben eines Formulars zusätzliche Daten an ein CGI-Programm schicken?
8. Wie kann ein CGI-Programm den Browser zu einer bestimmten Webseite leiten?

## Übungen

1. In Übung 1 aus Kapitel 13 haben Sie ein Umfrage-Formular zum Thema »internetfreier Sonntag« erstellt. Zu diesem Formular sollen Sie nun ein passendes CGI-Programm aufsetzen. Um die Aufgabe nicht unnötig zu komplizieren, soll das CGI-Programm nur die Pro/Kontra-Optionsfelder auswerten und den Stand der Umfrage aktualisieren. Den Stand der Umfrage speichert man am besten in einer Datei. Beim Start des CGI-Programms lädt es diese Datei und liest den alten Stand ein. Dann aktualisiert das Programm den augenblicklichen Stand der Umfrage auf der Grundlage der vorliegenden Formulardaten und speichert den neuen Stand wieder in der Datei. Zum Schluss soll das Programm eine Webseite mit dem aktuellen Stand der Umfrage zurückliefern.

Noch ein Hinweis zur Auswertung der Optionsfelder: Der Browser liefert als Wert für eine Gruppe von Optionsfeldern (gleiches name-Attribut) den value-Wert des Optionsfeldes, das gesetzt ist.

---

❖ Kapitel Inhalt Index **SAMS** ❖ Top Kapitel ❖

---

1  
Perl wird wie JavaScript interpretiert. Der Interpreter muss dazu aber extra auf dem Rechner, auf dem die Perl-Programme ausgeführt werden sollen, installiert werden.

2  
Je nach Apache-Version kann das Verzeichnis auch /usr/local/apache/cgi-bin/ lauten. Sie können aber auch ein eigenes Verzeichnis für Ihre CGI-Programme festlegen.

3  
Wenn das aktuelle Verzeichnis nicht im Ausführungspfad eingetragen ist, müssen Sie es explizit mit angeben: ./hallo.pl.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

## Tag 18

# Active Server Pages

Heute wollen wir Sie in die serverseitige Programmierung à la Microsoft einführen. ASP steht für »Active Server Pages«, also aktive Server-Seiten, und der Name ist Programm. Bei den Webservern von Microsoft ist die ASP-Unterstützung im Lieferumfang mit dabei, und sofern Sie Microsoft Windows 95, 98, Me, 2000 oder NT einsetzen, können Sie loslegen. Die zugehörige Software finden Sie - sofern nicht eh bereits installiert - auf Ihrer Betriebssystem-CD oder können Sie im Internet herunterladen. Im Anhang finden Sie eine etwas ausführlichere Installationsanleitung. Stellen Sie sicher, dass Ihr lokaler Webserver auf die Unterstützung von ASP eingerichtet ist, damit Sie die heutigen Beispiele auch nachvollziehen sowie eigene Erfahrungen mit ASP sammeln können.

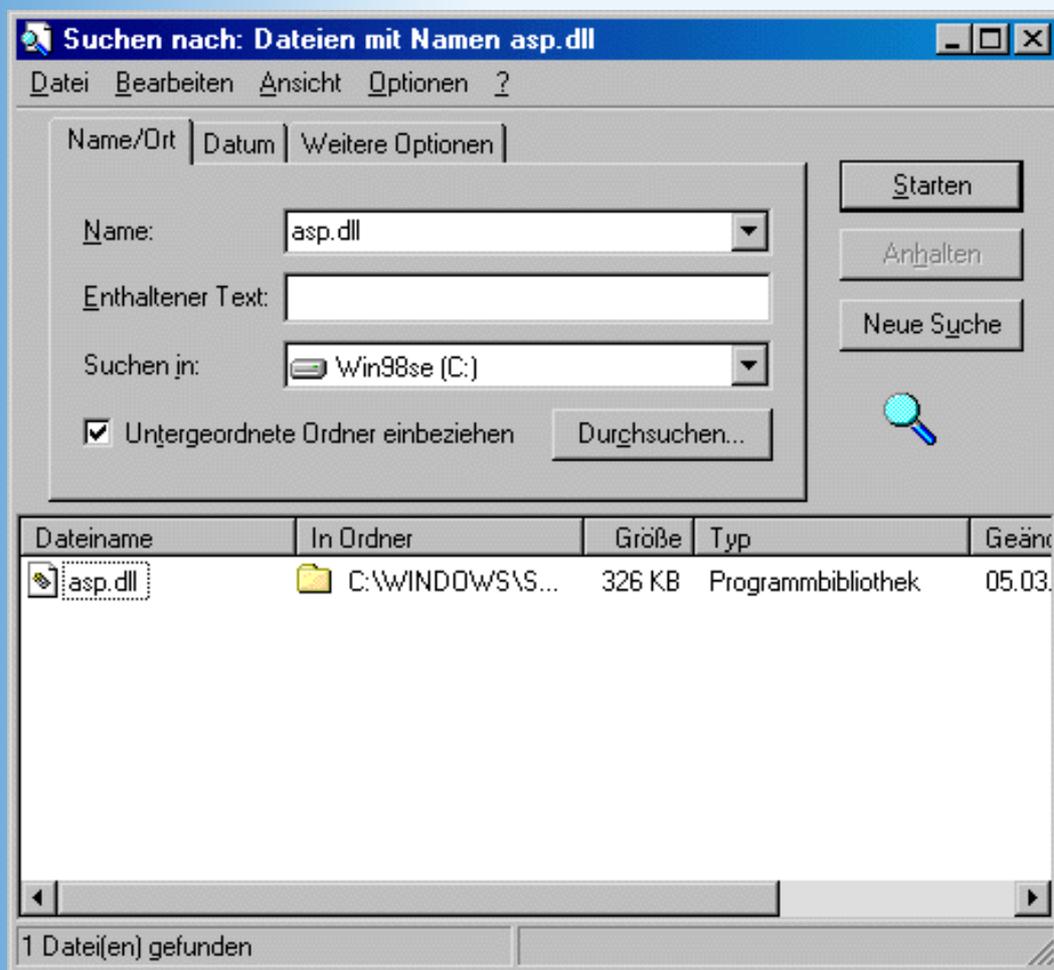
Beachten Sie bitte auch, dass Sie heute nur eine Einführung in ASP erhalten, Sie aber in der begrenzten Zeit keinen vollständigen Kurs erhalten können. Für weitergehende Informationen sowie eine deutlich ausführlichere Abhandlung empfehlen wir Ihnen aus dem Verlagsprogramm den Titel »Jetzt lerne ich Active Server Pages Programmierung«.

Die Themen heute:

- Wie ASP funktioniert
- Mit Variablen arbeiten
- Operatoren
- Schleifen und Fallunterscheidungen
- Datumswerte
- Formulare
- Mit Dateien arbeiten
- Cookies

## 18.1 Wie funktioniert eigentlich ASP?

Im Einleitungsabschnitt haben Sie es bereits gelesen: ASP muss bei Ihnen funktionsfähig installiert sein. Machen Sie nun Folgendes: Öffnen Sie mit **Start/Suchen/Dateien/ Ordner** den Dialog zum Durchsuchen Ihrer Festplatte nach Dateien, und suchen Sie dort nach »asp.dll«. Nach einiger Zeit sollten Sie fündig werden; die entsprechende Datei liegt in der Regel im Verzeichnis `Windows\System\inetsrv` bzw. `Windows\System32\inetsrv`.

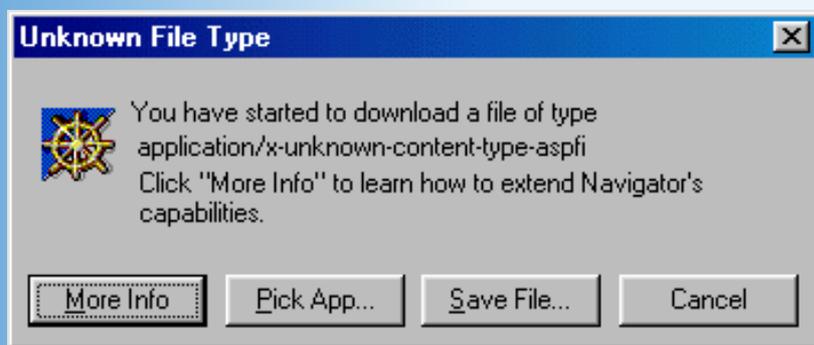


**Abbildung 18.1: Die Datei wurde gefunden**

Diese Datei, weniger als 400 KB groß, soll alles sein, was sich hinter ASP verbirgt? Nun ja, ein Fünkchen Wahrheit steckt da schon dahinter. Versetzen Sie sich in der Lage eines Webserver.

Wenn wir heute von einem Webserver reden, meinen wir damit einen Webserver mit ASP-Unterstützung, also in der Regel einen Internet Information Server (IIS) oder einen Personal Web Server (PWS), jeweils von Microsoft. Von diesem Server also wird eine ASP- Seite angefordert. Der Webserver weiß nun, dass er die Datei *asp.dll* zu Rate ziehen muss, denn diese interpretiert die ASP- Seite und führt den auf dieser Seite aufgeführten Code aus. Die Rückgabe der DLL ist dann (im Regelfall) HTML-Code, denn der Browser kann nur diesen interpretieren, mit ASP-Kommandos kommt er nicht klar.

Achten Sie also bei Ihren Tests unbedingt darauf, dass Sie den Server die ASP-Seiten ausführen lassen. Rufen Sie Ihre Beispielskripte via *http://servername/skriptname.asp* auf, und nicht über das Dateisystem, beispielsweise *c:\inetpub\wwwroot\skriptname.asp*. In Abbildung 18.2 sehen Sie, wozu das führen kann: Der Netscape Communicator öffnet einen Dialog zum Abspeichern der Datei, der Internet Explorer bietet dasselbe an, oder öffnet die Datei in einem Editor wie etwa Microsoft Frontpage.



**Abbildung 18.2: Netscape Navigator will die ASP-Datei speichern**



*Sie müssen die Suche natürlich auf dem Rechner ausführen, auf dem der ASP-fähige Webserver läuft!*

*Der Begriff »ASP« geisterte in den letzten Wochen und Monaten mehrmals durch die Medien und Sie sind vielleicht auch selbst ein paar Mal darauf gestoßen.*

*Mit Active Server Pages hatten die entsprechenden Artikel jedoch nur in den seltensten Fällen zu tun. Meistens ging es um **Application Service Providing**, wobei es sich - stark vereinfacht - um das Anbieten von Softwareapplikationen übers Internet handelt. Sie können also bei einem **Application Service Provider** (das - wie Application Service Providing auch - gerne mit **ASP** abgekürzt wird) die Nutzung einer speziell zugeschnittenen Textverarbeitung einkaufen, und dann im Minutentakt tippen. Da liegen Parallelen zu Abgabeterminen von Büchern sehr nahe...*

Bevor es nun los geht, sollten Sie ein letztes Mal Ihre ASP-Installation auf Funktionstüchtigkeit überprüfen. Speichern Sie folgenden Code als ASP-Datei ab (oder nehmen Sie die Version auf der CD), und rufen Sie diese Datei dann im Browser auf:

### **Listing 18.1: willkommen.asp**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>ASP-Test</title>
</head>
<body bgcolor="white">
<%
 Response.Write "<h1>"
 Response.Write "Willkommen bei ASP!"
 Response.Write "</h1>"
%>
</body>
</html>
```

Wenn Sie alles richtig gemacht haben, sollte das Ergebnis so aussehen wie in Abbildung 18.3. Falls nicht, lesen Sie noch einmal die Hinweise in diesem Kapitel sowie die Installationshinweise im Anhang!

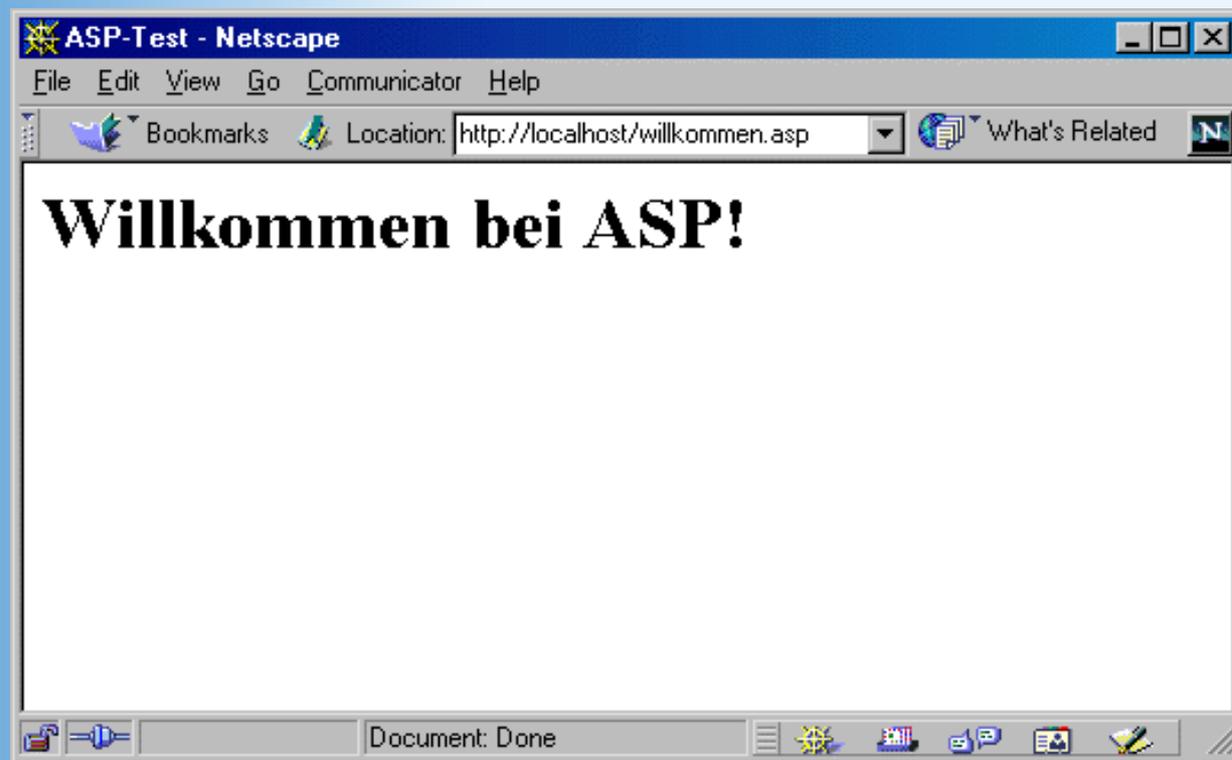


Abbildung 18.3: ASP ist korrekt eingerichtet worden!

## 18.2 Spracheinführung

Auf den nun folgenden Seiten erhalten Sie eine Schnelleinführung in ASP. Die verschiedenen Konzepte kennen Sie bereits aus der zweiten Woche, in der Sie JavaScript genauestens kennen gelernt haben. Sehen Sie uns also nach, wenn wir uns hier und da etwas kurz fassen, denn zum einen wollen wir Ihre (und unsere) Zeit nicht mit Redundanzen und Wiederholungen verschwenden, und zum anderen wollen wir schnellstmöglich zu den Anwendungsmöglichkeiten von ASP vordringen, denn das ist deutlich interessanter als die trockene Theorie.

Zunächst aber noch ein Wort zur Programmiersprache. ASP selbst ist keine Programmiersprache, sondern eine Technologie bzw. ein Grundgerüst, die/das es ermöglicht, auf Microsoft-Servern serverseitig zu programmieren. Sie haben in Bezug auf die verwendete Programmiersprache eine Reihe von Wahlmöglichkeiten, und wir haben uns hier für Visual Basic Script (kurz: VBScript) entschieden. Warum? Hier sind ein paar der Gründe:

- VBScript unterscheidet nicht zwischen Groß- und Kleinschreibung und ist dadurch etwas einsteigerfreundlicher als andere Sprachen.
- VBScript ist keine allzu mächtige Sprache. Für Profis mag das ein Nachteil sein, aber für Einsteiger sind das natürlich die besten Voraussetzungen, da man sehr schnell einen Überblick über den Sprachschatz erlangen kann.
- VBScript ist der De-facto-Standard für die ASP-Programmierung im World Wide Web. Das heißt: Wenn Sie im Internet nach kostenlosen ASP-Skripten suchen, werden Sie diese in VBScript programmiert finden. Wenn Sie Leute um Hilfe fragen (beispielsweise in Newsgroups), so werden die meisten Leute ihre ASP-Seiten auch mit VBScript programmieren.

Hauptkonkurrenz zu VBScript ist JScript. JScript ist das Microsoft-Pendant zu JavaScript. Wenn Sie also im Internet Explorer JavaScript-Programme ausführen, dann werden die dort von der JScript-Engine interpretiert. Sie werden jetzt vielleicht die nicht ganz unberechtigte Frage stellen, wieso wir dann an dieser Stelle nicht JScript verwenden, wenn Sie (der Leser) nach der Lektüre der Tage 8 bis 13 bereits JavaScript-Profis sind? Nun, drei der Gründe finden Sie oben aufgeführt, und es gibt noch einen dritten Grund: JScript ist zwar *ziemlich* kompatibel zu JavaScript, aber eben nicht ganz. Vor allem bei der serverseitigen Programmierung stößt man hier recht schnell auf Hindernisse.

## Sprung ins kalte Wasser

Werfen Sie noch einmal einen Blick auf oben bereits aufgeführten Programmcode:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>ASP-Test</title>
</head>
<body bgcolor="white">
<%
 Response.Write "<h1>"
 Response.Write "Willkommen bei ASP!"
 Response.Write "</h1>"
%>
</body>
</html>
```

Sie sehen zunächst einmal eine Reihe von HTML-Code. Etwa in der Mitte finden Sie ein paar Ihnen bis dato eher unbekannt Anweisungen. Sie werden durch `<%` und `%>` begrenzt. Daraus können Sie eine erste Schlussfolgerung ableiten: ASP-Kommandos werden durch `<%` und `%>` begrenzt.



*Wenn wir im Folgenden von ASP-Code und ASP-Kommandos reden meinen wir streng genommen VBScript-Code/-Kommandos, der/die auf einer ASP-Seite integriert ist/sind.*

Werfen Sie als Nächstes einen Blick auf die drei Zeilen eigentlichen ASP-Code: Sie haben alle den folgenden Aufbau:

```
Response.Write "irgendein Text"
```

Es sollte klar sein, was hier passiert: Der hinter `Response.Write` angegebene Text wird ausgegeben und an den Browser geschickt. Und in der Tat, wenn Sie sich in Ihrem Webbrowser den Quelltext der aktuellen Seite anzeigen lassen (Internet Explorer: **Ansicht/Quelltext anzeigen**, Netscape Navigator: **View/Page Source** oder **Ansicht/Seitenquelltext**), sehen Sie Folgendes:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>ASP-Test</title>
</head>
<body bgcolor="white">
<h1>Willkommen bei ASP!</h1>
</body>
</html>
```

Die drei Texte (oder, um bei Programmiersprachentermini zu bleiben, Strings/ Zeichenketten) wurden also tatsächlich wie gewünscht ausgegeben.

Was das mit `Response.Write` genau auf sich hat, werden wir an späterer Stelle genauer erläutern. Zunächst nur so viel: ASP bietet eine Reihe von so genannten *Objekten* an, welche erweiterte Funktionalitäten bieten. Das `Response`-Objekt ist eines dieser Objekte, und es hat eine Methode namens `Write`, welche Texte an den Browser ausgibt. Eine Methode ist in etwa so etwas wie eine Funktion, ein Konzept, das Sie bereits in den JavaScript-Kapiteln kennen gelernt haben.



*»Aber bei JavaScript werden Funktionsparameter doch geklammert?!« - Sie haben recht, das ist in der Tat ein Unterschied. Aber zur Beruhigung: Sofern Sie nur einen Parameter übergeben, können Sie auch Klammern verwenden; `Response.Write "irgendein Text"` funktioniert also auch.*

## Variablen

Wie in jeder anderen Programmiersprache auch, dreht sich bei ASP vieles, wenn nicht sogar fast alles, um Variablen. Variablen können dazu verwendet werden, Daten zwischenspeichern, Daten miteinander zu verbinden, und vieles mehr.

## Zuweisungen

Eine Zuweisung an eine Variable geht in VBScript wie folgt vonstatten:

```
<%
Option Explicit
Dim ping
ping = "pong"
%>
```

Mittels `Dim ping` wird eine Variable namens `ping` definiert, und dieser dann eine Zeile weiter die Zeichenkette `"pong"` als Wert zugewiesen. Doch wozu dient das `Option Explicit`?

Um dies herauszufinden, betrachten Sie einmal das folgende Skript, welches nur leicht abgeändert

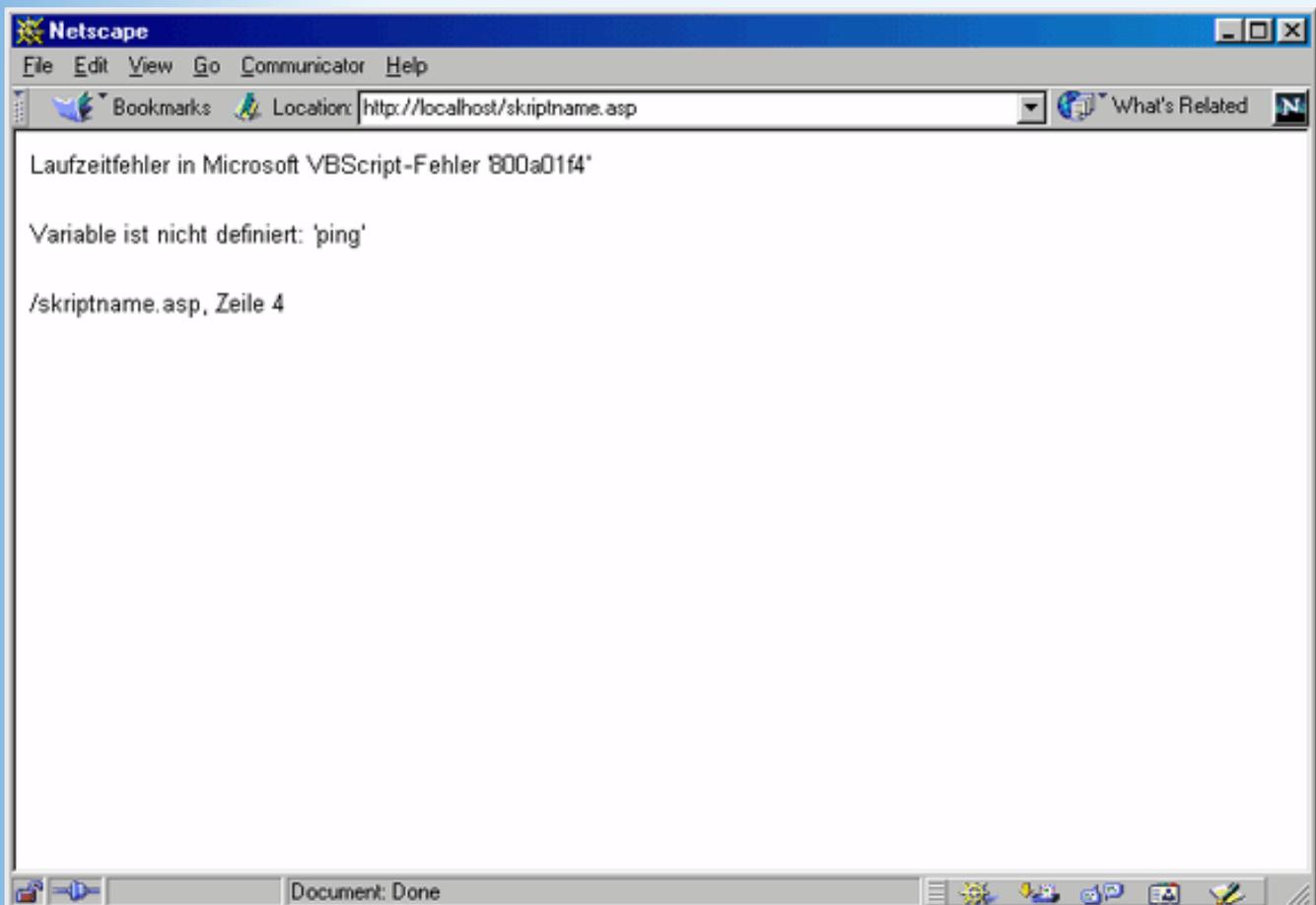
worden ist:

```
<%
Option Explicit
' Dim ping
ping = "pong"
>
```

Die Zeile Dim Ping beginnt jetzt mit einem Apostroph/Hochkomma (auf der Tastatur: (Umschalt)+(#)). Hierdurch wird ein einzeiliger Kommentar eingeleitet, vergleichbar mit // in JavaScript-Code. Hierdurch wird der Befehl Dim ping also vor dem ASP- Interpreter versteckt, und damit die Variable ping nicht mehr explizit deklariert. Wenn Sie dieses Codestück in Ihrem Browser laden, erscheint eine (serverseitig generierte) Fehlermeldung (siehe Abbildung 18.4).



*Anstelle eines Apostrophs können Sie einen Kommentar auch mit Rem einleiten (kurz für **Remark**, engl. Anmerkung).*



**Abbildung 18.4: Die Variable wurde nicht definiert**

Die Variable wurde also nicht deklariert. Sie erinnern sich nun vielleicht an die Argumentation von zuvor, dass wir VBScript gegenüber JScript unter anderem wegen der Einsteigerfreundlichkeit den Vorzug gegeben haben. Bei JScript jedoch kann eine Variable gut und gerne mit var variablenname

deklariert werden, muss aber nicht. Nun, die gute Nachricht ist: Bei VBScript ist das nicht großartig anders. Auch hier können Sie Variablen ohne explizite Deklaration verwenden, Sie benötigen also überhaupt kein Dim.

## Deklarieren oder nicht?

Im Gegensatz zu JavaScript bietet VBScript auch noch einen Modus an, der den Programmierer dazu zwingt, Variablen zu deklarieren. Dieser Modus wird mit Option Explicit aktiviert. Wenn Sie diese Anweisung einsetzen, müssen Sie auf dieser Seite alle verwendeten Variablen mittels Dim deklarieren, bevor Sie sie verwenden können.



*Option Explicit muss die allererste Anweisung in der ASP-Seite sein, muss also insbesondere vor allen anderen Kommandos sowie vor jedweder HTML-Ausgabe kommen!*

Warum das von Vorteil sein kann, sehen Sie an folgendem Beispiel. Dieses Beispiel stammt - auch wenn man das kaum glauben mag - aus der Praxis. Natürlich ist es stark vereinfacht, aber wie oft konnten wir auf die Fragen eines Kollegen, der den Fehler in seinem ASP-Code nicht finden konnte, antworten mit: Du hast Dich bei dem Variablennamen vertippt. Untenstehender Code ist also falsch - und wird Option Explicit nicht verwendet, ist die Fehlersuche (vor allem bei längeren Listings) sehr, sehr mühsam.

### Listing 18.2: optionexplicit.asp

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Option Explicit</title>
</head>
<body bgcolor="white">
<%
 Dim beispielebegruessung
 beispielebegruessung = "Guten Tag!"
 Response.Write "<h1>"
 Response.Write beispielebegruessung
 Response.Write "</h1>"
%>
</body>
</html>
```

Der Browser gibt nichts aus, wovon Sie sich selbst überzeugen können, wenn Sie das Listing in Ihren Browser laden. Wenn Sie aber Option Explicit verwenden (als erste Anweisung, versteht sich, also noch vor <!DOCTYPE und <html>), wird der Fehler offensichtlich (siehe Abbildung 18.5).

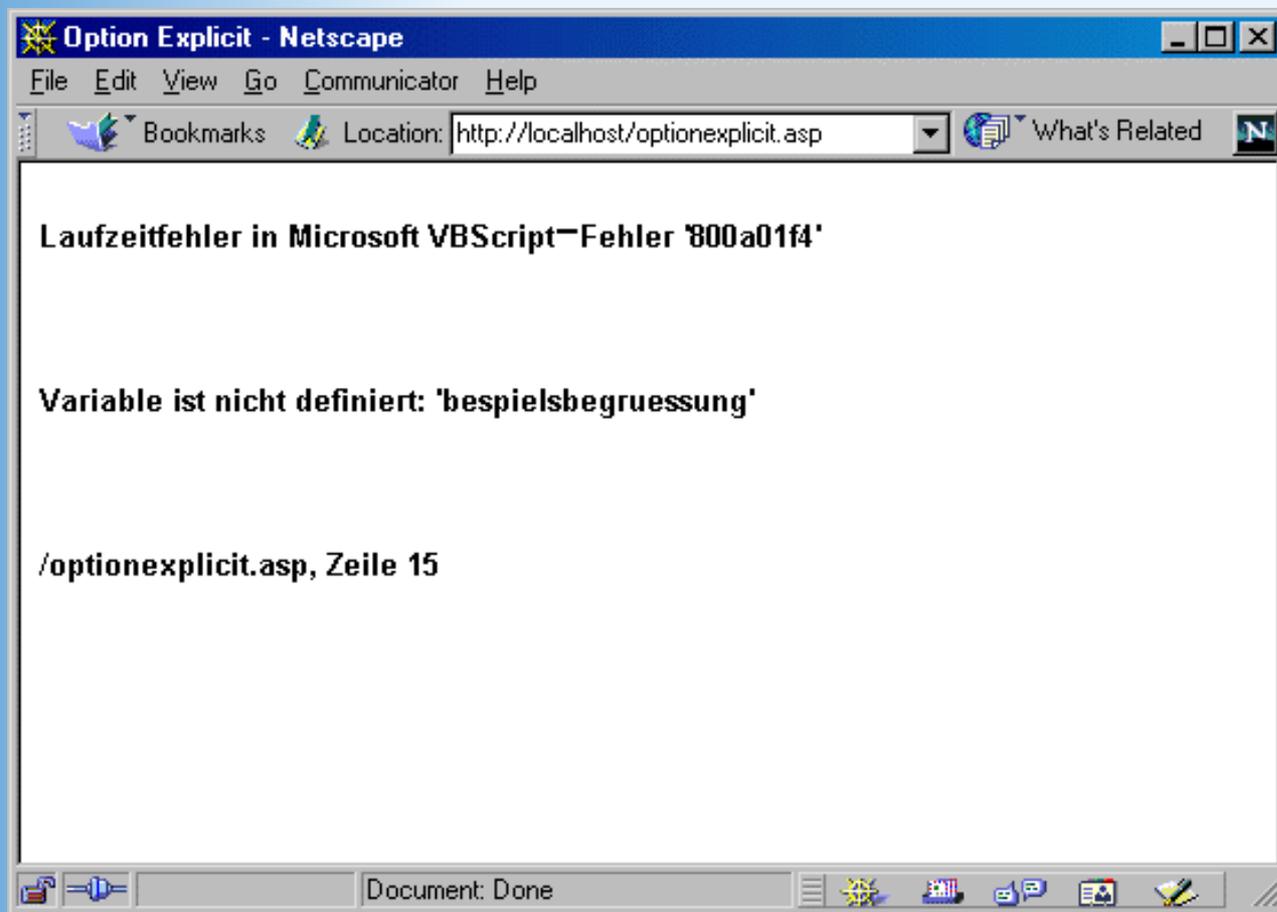


Abbildung 18.5: Ein Tippfehler im Variablennamen führt zur Fehlermeldung

## Ausgabe

Wie Sie Variablen ausgeben können, haben Sie in obigem Beispiel bereits gesehen: Sie verwenden einfach `Response.Write` und übergeben als Parameter (mit oder ohne Klammern, das ist in diesem Fall wie gesagt Ihnen überlassen) den Variablennamen. Hier ein Beispiel mit mehreren Variablen:

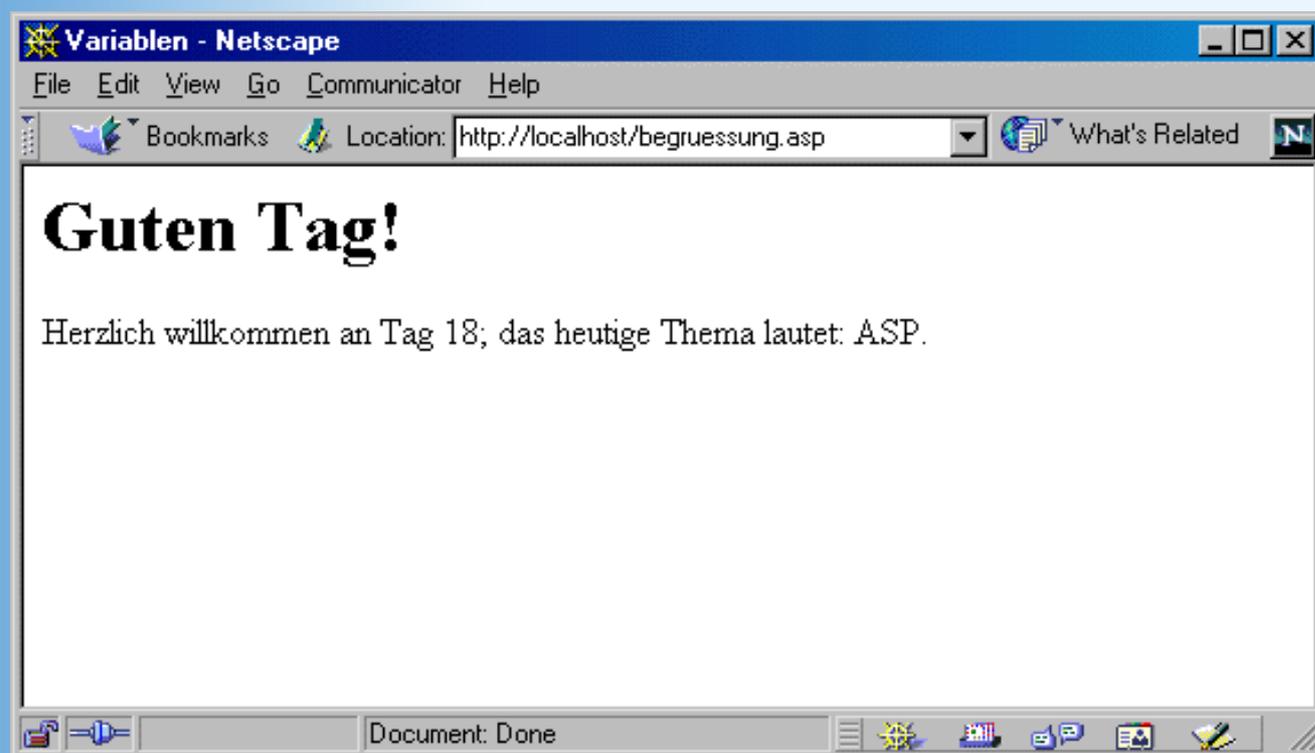
### Listing 18.3: begrueessung.asp

```
<%
 Option Explicit
 Dim begrueessung, thema
 Dim tag
 begrueessung = "Guten Tag!"
 tag = 18
 thema = "ASP"
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Variablen</title>
</head>
<body bgcolor="white">
<h1>
<%
```

```

Response.Write begruessung
%>
</h1>
Herzlich willkommen an Tag
<%
 Response.Write tag
%>;
das heutige Thema lautet:
<%
 Response.Write thema
%>.
</body>
</html>

```



**Abbildung 18.6: Der Inhalt der Variablen wird ausgegeben**

Ihnen sollten an obigem Code auch noch zwei Dinge auffallen:

- Sie können mit einer Dim-Anweisung gleich mehrere Variablen deklarieren, wenn Sie dazu die Variablennamen durch Kommata voneinander trennen.
- Sie können auf einer ASP-Seite umgekehrt auch mehrere Dim-Anweisungen (natürlich für verschiedene Variablen) verwenden.

Der Code wird nun doch schon etwas lang, obiger Code etwa umfasst circa eine Druckseite. Für die spartanischen Informationen, die auf der resultierenden Seite angezeigt werden, ist das schon ein wenig zu viel des Guten. Es gibt jedoch zwei Möglichkeiten, den Code etwas zu straffen:

- Verzichten Sie bei kurzen Codeanweisungen auf die Zeilensprünge nach `<%` und vor `%>`:

```
<% Response.Write "In der Kürze liegt die Würze" %>
```

- Verwenden Sie die folgende Kurzform zur Ausgabe von Variablen und Strings:

<% =Variablenname %> oder <% ="Noch kürzer" %>

Diese Kurzform darf nur als einzige Anweisung innerhalb eines <%-%>-Blocks stehen!

Obiger Code lässt sich also etwas kürzer fassen:

#### Listing 18.4: begruessung\_kurz.asp

```
<%
Option Explicit
Dim begruessung, thema
Dim tag
begruessung = "Guten Tag!"
tag = 18
thema = "ASP"
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Variablen</title>
</head>
<body bgcolor="white">
<h1><% =begruessung %></h1>
Herzlich willkommen an Tag <% =tag%>;
das heutige Thema lautet: <% =thema %>.
</body>
</html>
```

Es gibt auch noch eine weitere Abkürzung, die jedoch in der Regel auf Kosten der Übersichtlichkeit geht. Sie können mehrere Anweisungen in eine Zeile schreiben, wenn Sie diese Anweisungen durch einen Doppelpunkt voneinander trennen. Die Variablenzuweisungen in obigem Listing könnten Sie also auch wie folgt etwas kürzer fassen:

```
<%
begruessung = "Guten Tag!" : tag = 18 : thema = "ASP"
%>
```

Auf der anderen Seite ist es ebenfalls möglich, eine relativ lange Anweisung in mehrere Zeilen aufzuspalten. Betrachten Sie einmal folgendes Beispiel. Es kommt im Satz zu einem erzwungenen Zeilenumbruch, weil die Anweisung (aufgrund der überdimensionalen Zeichenkette) länger als eine Zeile ist:

```
<%
Response.Write "Kant sagte einst: Schön ist dasjenige, was ohne Interesse
gefällt"
%>
```

Aus Autorensicht ist so ein Umbruch natürlich etwas sehr unerwünschtes. Erkennt ein Leser beispielsweise nicht, dass der Umbruch aus rein satztechnischen Gründen geschehen ist (auch ein vom Setzer eingefügtes Symbol, dass es sich hier um einen solchen Umbruch handelt hilft hier oft nicht viel), kann es sein, dass bei ihm der Code nicht funktioniert. Aus diesem Grund behelfen auch wir uns des öfteren mit der Möglichkeit, längere Codezeilen aufzuteilen. Wie das geht? Mit dem Unterstrich \_ (Tastatur: (Umschalt)+(-)) kann man lange Zeilen auftrennen - aber nur zwischen einzelnen Teilen eines Kommandos, nicht beispielsweise innerhalb von Zeichenketten.

```
<%
 Response.Write _
 "Kant sagte einst: Schön ist dasjenige, was ohne Interesse gefällt"
>
```

Dadurch gewinnt der Autor so viel an Platz, dass er die nachfolgende(n) Zeile(n) oft sogar noch ein wenig einrücken kann, aus rein ästhetischen Gründen.

## Variablentypen und Namensgebung

Es gibt in ASP eine Reihe von Variablentypen, die gebräuchlichsten sind jedoch die Folgenden:

- Zeichenketten/Strings:

```
Dim strName : strName = "Ich bin kein Berliner"
```

- Integerwerte (ganze Zahlen):

```
Dim intName : intName = 42
```

- Doublewerte (Fließkommazahlen):

```
Dim dblName = 3.141592654
```

- Booleans (Wahrheitswerte):

```
Dim bName : bName = True : bName = False
```

Weitere Variablentypen stellen wir an den entsprechenden Stellen am heutigen Tag noch vor.



*Sie können die unterschiedlichen Variablentypen auch ineinander umwandeln, also beispielsweise aus einer numerischen Variablen eine Stringvariable machen. Der Typ einer Variablen kann sich innerhalb eines ASP-Skripts jederzeit ändern.*

Beachten Sie in obiger Auflistung die Namensgebung der Variablen. Sie sind hier natürlich freigestellt (wie üblich müssen Variablennamen eindeutig sein, sollten am besten ausschließlich aus Buchstaben

bestehen, und insbesondere kein Minus-Zeichen enthalten), aber auch hier haben sich einige Regeln eingebürgert. Vor dem eigentlichen Variablennamen (der mit einem Großbuchstaben beginnen sollte) wird ein kurzes Präfix gestellt, das den Typ der Variablen angibt. Beispielsweise lautet eine Empfehlung, dass die Namen von Stringvariablen mit str beginnen, Boolean-Variablen mit b, und so weiter. Auch wenn das insbesondere für den deutschen Leser etwas ungewohnt aussieht - englische Präfixe, gefolgt von einem unter Umständen deutschen Variablennamen - so hält es doch den Code nachvollziehbar, insbesondere wenn einmal jemand anderes Ihren Code überarbeiten muss.



*Wir werden uns in diesem Buch deswegen an diese Empfehlungen (um nicht zu sagen: an diese Quasi-Standards) halten, Sie sind aber natürlich in der Wahl Ihrer Variablennamen sowie der Namenskonventionen für Variablen frei. Sie sollten sich aber unbedingt dazu durchringen, eine Namenskonvention einzurichten und dann auch einzuhalten, um eine gewisse Form der Einheitlichkeit in Ihrem Code zu erhalten. Außerdem sind damit Stolperfallen der Art »steht in der Variablen jahr die Jahreszahl als Integer (2001) oder als zweistelliger String («01«)?« quasi ausgeschlossen.*

## Arrays

Selbstverständlich bietet Ihnen VBScript auch die Möglichkeit, mehrere Werte unter dem Oberbegriff einer einzelnen Variablen zu speichern: Die Rede ist natürlich von Arrays. Zum Einstieg aber gleich eine Warnung, die uns so ernst ist, dass wir sie in einen eigenen Absatz packen:



*Verwenden Sie Arrays mit VBScript so selten wie möglich!*

Wir meinen das auch ernst. Die Array-Behandlung von VBScript ist im Vergleich zu fast allen anderen Programmiersprachen indiskutabel. Sie sollten also Arrays tunlichst vermeiden, aber wie Sie sehen werden, benötigen Sie Arrays auch äußerst selten.

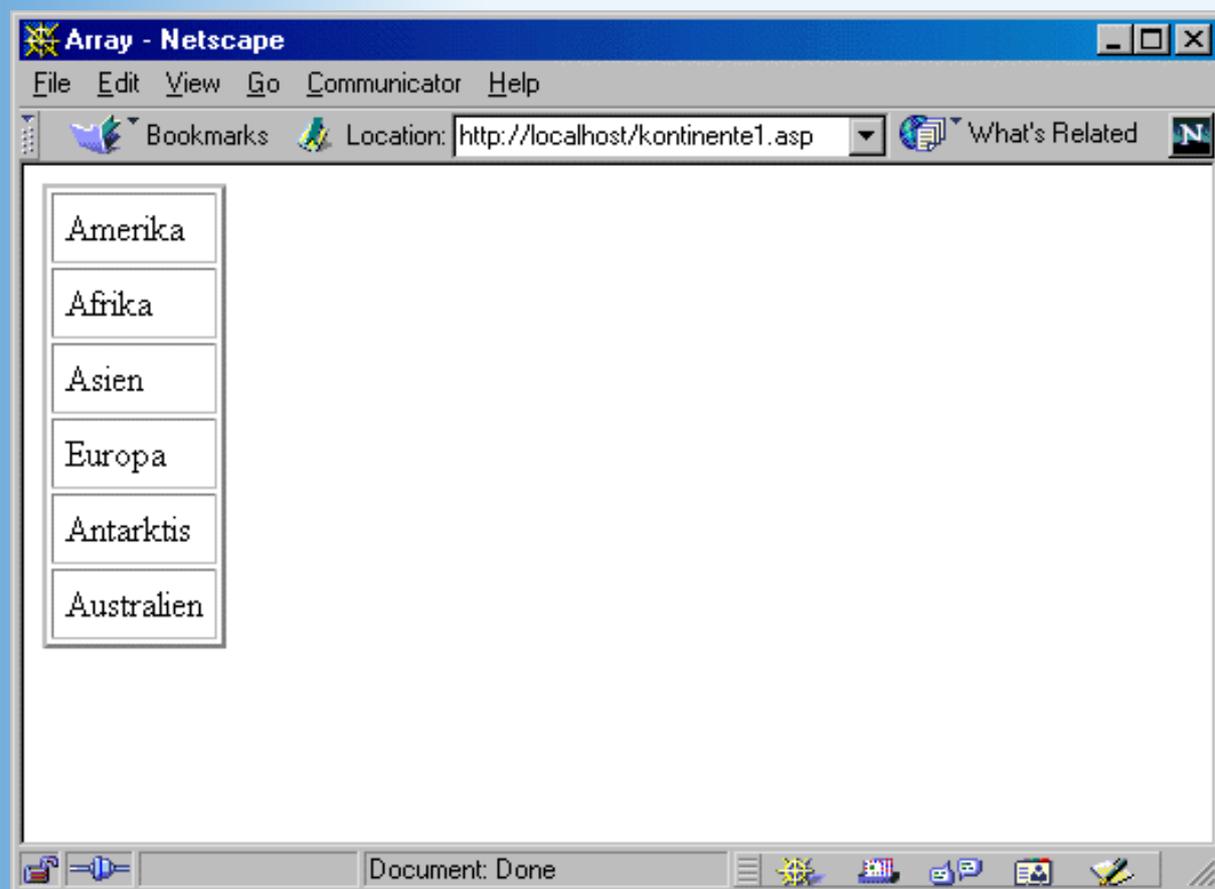
Ein Array deklarieren Sie wie jede andere Variable auch, also mit Dim. Sie müssen aber hinter dem Variablennamen in Klammern den Index des letzten Elements im Array angeben. Die Zählung der Elemente beginnt bei 0, mit Dim arystName(41) erstellen Sie also ein Array mit maximal 42 Elementen. Hier ein kleines Beispiel:

### Listing 18.5: kontinente1.asp

```
<%
Option Explicit
Dim arystName(5)
arystName(0) = "Amerika"
arystName(1) = "Afrika"
```

```
arystrKontinente(2) = "Asien"
arystrKontinente(3) = "Europa"
arystrKontinente(4) = "Antarktis"
arystrKontinente(5) = "Australien"
```

```
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Array</title>
</head>
<body bgcolor="white">
<table border="2" cellpadding="5">
 <tr><td><% =aryStrKontinente(0) %></td></tr>
 <tr><td><% =aryStrKontinente(1) %></td></tr>
 <tr><td><% =aryStrKontinente(2) %></td></tr>
 <tr><td><% =aryStrKontinente(3) %></td></tr>
 <tr><td><% =aryStrKontinente(4) %></td></tr>
 <tr><td><% =aryStrKontinente(5) %></td></tr>
</table>
</body>
</html>
```



**Abbildung 18.7: Die sechs Kontinente**

Auch hier wieder eine Empfehlung für die Namensgebung: Dass es sich bei einer Variablen um ein Array handelt, können Sie durch das Präfix `ary` kennzeichnen. Nun gibt es aber Arrays, die Zeichenketten enthalten, Arrays, die nur numerische Werte beinhalten, und so weiter. Sie sollten also

ein zweites Präfix nehmen, das den Datentyp der Arrayelemente angibt. Im obigen Beispiel haben wir das gemacht: Das Array enthält nur Zeichenketten, und hat deswegen das Präfix `arystr`.



*Arrays können verschiedene Datentypen enthalten, das ist dem VBScript-Interpreter völlig egal. Verwenden Sie dann bei der Namensgebung nur das Präfix `ary`.*

Wenn Sie anfangs noch nicht festlegen können (oder wollen), wie viele Elemente ein Array letztendlich enthält, können Sie auch ein *dynamisches Array* anlegen. Geben Sie dazu einfach bei der Deklaration keinen maximalen Index an:

```
<%
 Dim arystrKontinente()
%>
```

Sie können nun mittels der Anweisung `ReDim` die Anzahl der Elemente in dem Array im Nachhinein festlegen:

```
<%
 Dim arystrKontinente()
 ReDim arystrKontinente(6)
 arystrKontinente(0) = "Amerika"
 arystrKontinente(1) = "Afrika"
 arystrKontinente(2) = "Asien"
 arystrKontinente(3) = "Europa"
 arystrKontinente(4) = "Antarktis"
 arystrKontinente(5) = "Australien"
 arystrKontinente(6) = "Bayern"
%>
```

Nun stellen Sie Ihren Fehler fest und können - durch erneuten Aufruf von `ReDim` die Anzahl der Elemente im Array korrigieren (in diesem Fall nach unten). Hierbei gehen alle Elemente, die einen höheren Index haben als der neue Maximalindex, verloren. Wenn Sie so wollen: Bayern ist also kein eigener Kontinent mehr<sup>1</sup>.

```
<%
 Dim arystrKontinente()
 ReDim arystrKontinente(6)
 arystrKontinente(0) = "Amerika"
 arystrKontinente(1) = "Afrika"
 arystrKontinente(2) = "Asien"
 arystrKontinente(3) = "Europa"
 arystrKontinente(4) = "Antarktis"
 arystrKontinente(5) = "Australien"
 arystrKontinente(6) = "Bayern"
 ReDim arystrKontinente(5)
```

%>



*Auch, wenn Sie später noch einmal mittels ReDim das Array größer machen, erhalten Sie den Wert »Bayern« nicht mehr zurück, der ist unwiderruflich verloren.*

In Bezug auf Arrays noch eine letzte Funktion: Mittels UBound erhalten Sie den größtmöglichen Index des Arrays:

```
<%
Dim arystrKontinente()
ReDim arystrKontinente(6)
Response.Write "<p>Maximaler Index: "
Response.Write UBound(arystrKontinente) 'Ausgabe: 6
Response.Write "
"
ReDim arystrKontinente(5)
Response.Write "Neuer maximaler Index: "
Response.Write UBound(arystrKontinente) 'Ausgabe: 5
Response.Write "</p>"
%>
```

Zu guter Letzt noch ein kleiner Tipp: Sie können mit der Funktion Array() ein Array auch bequemer vorfüllen; übergeben Sie einfach die Arraywerte als Parameter:

```
<%
Dim arystrKontinente
arystrKontinente = Array("Amerika", "Afrika", "Asien", "Europa", _
 "Antarktis", "Australien")
%>
```

## Operationen und Operatoren

Auch diese Thematik ist eigentlich ein alter Hut und Ihnen aus dem JavaScript-Teil bereits bestens bekannt. Aus diesem Grund ersparen wir Ihnen an dieser Stelle alle weiteren langen Vorreden und steigen direkt in die Materie ein. Auch in VBScript unterscheidet man zwischen arithmetischen Operatoren, Vergleichsoperatoren, logischen Operatoren sowie Zeichenkettenoperatoren und -operationen.

### Arithmetische Operatoren

Die meisten arithmetischen Operatoren sind wohl noch aus der Schule bekannt - und da Sie ja inzwischen schon ein JavaScript-Halbgott sind, kennen Sie den größten Teil davon eh schon. In Tabelle 18.1 finden Sie eine Übersicht, sowie das jeweils entsprechende JavaScript-Äquivalent.

Operator	Beschreibung	Beispiel	JavaScript-Äquivalent
+	Addition	11 + 4 (= 15)	+
-	Subtraktion	11 - 4 (= 7)	-
*	Multiplikation	11 * 4 (=44)	*
/	Division	11 / 4 (=2.75)	/
Mod	Rest bei Division	11 Mod 4 (=3)	%
^	Exponentialrechnung	11 ^ 4 (= 14641)	pow()
\	Ganzzahlige Division	11 \ 4 (= 2)	(nicht vorhanden)

**Tabelle 18.1: Die arithmetischen Operatoren von VBScript**

An dieser Stelle ist es höchste Zeit, den Zuweisungsoperator = noch einmal offiziell einzuführen (auch wenn wir ihn zuvor schon mehrmals untergeschoben haben). Das ganze läuft hier exakt wie in JavaScript: Rechts vom Gleichheitszeichen steht ein Ausdruck, und der Wert dieses Ausdrucks wird der Variablen links vom Gleichheitszeichen zugewiesen:

```
<%
Option Explicit
Dim intTageProJahr, intAnzahlWochen, intTageProWoche
intAnzahlWochen = 52
intTageProWoche = 7
intTageProJahr = intAnzahlWochen * intTageProWoche + 1
' intTageProJahr hat jetzt den Wert 365
Response.Write "Ein (nicht Schalt-)Jahr hat "
Response.Write intTageProJahr
Response.Write " Tage"
%>
```

## Vergleichsoperatoren

Unter einem Vergleichsoperator versteht man genau das, was der Name schon verspricht - zwei »Dinge« (Variablen, Werte, Ausdrücke) werden miteinander verglichen. Das Ergebnis einer Vergleichsoperation ist ein boolescher Wert, also entweder True oder False. Sie benötigen Vergleichsoperationen für die diversen Kontrollstrukturen von VBScript, welche Sie in Kürze kennen lernen werden. Zunächst sollten Sie aber die Vergleichsoperationen kennen lernen, und auch dieses Mal ziehen wir wieder den Vergleich zu JavaScript:

Operator	Beschreibung	Beispiel	JavaScript-Äquivalent
>	größer als	2 > 1 (= True)	>
<	kleiner als	2 < 1 (= False)	<
>=	größer oder gleich als	2 >= 1 (= True)	>=
<=	kleiner oder gleich als	2 <= 1 (= False)	<=

=	gleich	2 = 1 (= False)	==
<>	ungleich	2 <> 1 (= True)	!=

**Tabelle 18.2: Die Vergleichsoperatoren von VBScript**

## Logische Operatoren

Arithmetische Operatoren werden auf Integer- und Double-Werte angewandt, Vergleichsoperatoren ebenso (und unter bestimmten Umständen auch auf Strings). Boolesche Werte mit Termen wie »größer« und »kleiner« fassen zu wollen, ist schon etwas unintuitiv. Für diese Zwecke gibt es aber vier logische Operatoren, die Sie Tabelle 18.3 entnehmen können.

Operator	Bedeutung	Beispiel	JavaScript-Äquivalent
And	(logisches) Und	True And False (= False)	&&
Or	(logisches) Oder	True Or False (= True)	
Not	Nicht (Negation)	Not True (= False)	!
Xor	Entweder oder	True Xor False (= True)	(nicht vorhanden)

**Tabelle 18.3: Die logischen Operatoren von VBScript**

## Zeichenkettenoperatoren und -operationen

Die wohl naheliegendste Operation, die man mit zwei Zeichenketten anstellen kann, ist die *Konkatenation*, das Aneinanderhängen von zwei oder mehr Zeichenketten. erinnern Sie sich noch an die etwas mühseligen Ausgaben von Zeichenketten in den vorangegangenen Beispielen? Dort sah das immer ungefähr folgendermaßen aus:

```
<%
 Dim begruessung
 begruessung = "Guten Tag!"
 Response.Write "<h1>"
 Response.Write begruessung
 Response.Write "</h1>"
%>
```

Recht viel Platz für wenig Ausgabe, finden Sie nicht? Folgendes sieht da schon etwas besser (und kürzer) aus:

```
<%
 Dim begruessung
 begruessung = "Guten Tag!"
 Response.Write "<h1>" & begruessung & "</h1>"
%>
```

Die meisten der bisherigen Operatoren bestanden aus einem oder zwei (Sonder-)Zeichen (zum

Beispiel +, -, >=, <=), oder aus einem mehr oder minder kurzen Begriff (And, Or, ...). Alle weiteren Stringoperationen passen nicht genau in dieses Schema, denn es handelt sich hierbei um Funktionen, die aber mit Strings arbeiten.

Das Funktionen-Konzept kennen Sie ebenfalls bereits aus der JavaScript-Woche. Eine Funktion ist ein eigenständiges Funktionsstück, dem Sie in den meisten Fällen einen oder mehrere Parameter übergeben. Die Funktion hat dann (ebenfalls nur in den meisten Fällen) einen Rückgabewert, den Sie für die Programmierung benötigen.

In Tabelle 18.4 finden Sie nun eine Reihe von String-Operationen, mit denen Sie beispielsweise Teilbereiche aus einem String heraustrennen können.



*Die Funktionen haben alle eins gemeinsam: Sie ändern den übergebenen String in keinsten Weise, Sie müssen also immer den Rückgabewert verwenden!*

Funktion	Beschreibung	Parameter	Beispiel
Len(s)	Ermittelt die Länge eines Strings	s - der betrachtete String	Len(»M+T«) (= 3)
InStr(s1, s2)	Ermittelt die <i>erste</i> Position eines Teilstrings in einem längeren String. Der Rückgabewert 0 bedeutet, dass der Teilstring nicht im längeren String enthalten war.	s1 - der zu durchsuchende String  s2 - der zu suchende String	InStr(»abcbcd«, »bc«) (= 2)
InStrRev(s1, s2)	Ermittelt die <i>letzte</i> Position eines Teilstrings in einem längeren String. Der Rückgabewert 0 bedeutet, dass der Teilstring nicht im längeren String enthalten war.	s1 - der zu durchsuchende String  s2 - der zu suchende String	InStr(»abcbcd«, »bc«) (= 4)
Left(s, i)	Liefert die ersten Zeichen eines Strings zurück.	s - der betrachtete String  i - Anzahl der zu ermittelnden Zeichen	Left(»Markt+Technik«, 5) (= »Markt«)

Mid(s, a, b)	Liefert ein Teilstring aus einem String zurück.	s - der betrachtete String  a - Startposition des Teilstrings  b - Anzahl der zu ermittelnden Zeichen	Mid(»Markt+Technik«, 5, 3) (= »t+T«)
Right(s, i)	Liefert die letzten Zeichen eines Strings zurück.	s - der betrachtete String  i - Anzahl der zu ermittelnden Zeichen	Right(»Markt+Technik«, 7) (= »Technik«)
Replace(s, s1, s2)	Ersetzt in einem String einen Teilstring durch einen anderen und liefert den ersetzten String zurück.	s - der betrachtete String  s1 - der zu ersetzende Teilstring  s2 - der Teilstring, durch den ersetzt werden soll	Replace(»Markt & Technik«, » &«, » + «) (= »Markt + Technik«)

**Tabelle 18.4: Die in VBScript eingebauten Zeichenkettenfunktionen**

## Fallunterscheidungen

Programmierung ist nie linear. Je nach den äußeren Umständen (Benutzereingaben, dem aktuellen Datum, dem verwendeten Browser des Benutzers) muss ein serverseitiges Skript andere Aktionen ausführen.

## If-Anweisung

Haupt-Sprachkonstrukt von VBScript ist die If-Anweisung, die Sie - zumindest in ähnlicher Form schon von JavaScript her kennen. Sie haben an dieser Stelle aber zwei Hauptunterschiede, die beide (vor allem für den Einstieg) vorteilhaft sind:

- Die Bedingung bei der If-Anweisung muss nicht innerhalb von runden Klammern stehen - einer der Hauptfehler von angehenden JavaScript-Profis
- Es gibt mehrere Varianten des Else-Anweisung, der Code wird dadurch übersichtlicher und der Programmierer ist flexibler.

Um das ganze einmal anhand eines kleinen Beispiels darstellen zu können, hier ein kleiner Blick in die Zukunft: Mit Weekday(Now) können Sie den aktuellen Wochentag ermitteln. Dabei gibt die Funktion einen Rückgabewert zwischen 1 und 7 zurück, die die in Tabelle 18.2 ersichtliche Bedeutung haben:

Rückgabewert	Wochentag
1	Sonntag
2	Montag
3	Dienstag
4	Mittwoch
5	Donnerstag
6	Freitag
7	Samstag

**Tabelle 18.5: Die Rückgabewerte von Weekday**

Jetzt können Sie beispielsweise eine Warnmeldung ausgeben, wenn der Benutzer Ihr ASP- Skript an einem Sonntag aufruft:

```
<%
 Option Explicit
 Dim intTag
 intTag = Weekday(Now)
 If intTag = 1 Then
 Response.Write "Am Sonntag wird nicht gearbeitet!"
 End If
%>
```

Sie sehen hier schon den groben Aufbau einer If-Anweisung:

```
<%
 If Bedingung Then
 Befehlsblock
 End If
%>
```

Es gibt auch eine Kurzfassung, die sich aber nur dann lohnt, wenn der Befehlsblock einzeilig ist (oder durch : einzeilig gemacht werden kann):

```
<%
 If Bedingung Then Befehlsblock
%>
```

Sie können in diesem Fall also das End If weglassen - aber hüten Sie sich davor, denn dadurch wird der Code nur unwesentlich kürzer, aber wesentlich inkonsistenter!

## Else-Anweisung

Obiges Codebeispiel gibt nur sonntags etwas aus, an allen anderen Tagen bleibt die Seite leer. Höchste Zeit, dies zu beheben:

```

<%
Option Explicit
Dim intTag
intTag = Weekday(Now)
If intTag = 1 Then
 Response.Write "Am Sonntag wird nicht gearbeitet!"
Else
 Response.Write "Guten Tag, frisch ans Werk!"
End If
%>

```

Aus obigem Listing ist die Syntax der Else-Anweisung leicht ersichtlich:

```

<%
If Bedingung Then
 Befehlsblock
Else
 Befehlsblock
End If
%>

```



*Auch hier kann wieder das End If weggelassen werden, wenn alles in einer Zeile steht, aber wie wir bereits oben bemerkt haben, sollten Sie nicht an der falschen Stelle an Arbeit sparen.*

Streng genommen sollte man ja auch samstags nicht arbeiten. Ein erster Ansatz hierfür sieht so aus:

```

<%
Option Explicit
Dim intTag
intTag = Weekday(Now)
If intTag = 1 Then
 Response.Write "Am Sonntag wird nicht gearbeitet!"
Else
 If intTag = 7 Then
 Response.Write "Am Samstag wird nicht gearbeitet"
 Else
 Response.Write "Guten Tag, frisch ans Werk!"
 End If
End If
%>

```



Sanfte Linderung verspricht hier die Anweisung Elseif, welche eine Kombination aus Else und If ist, aber kein zusätzliches End If benötigt:

```
<%
 If Bedingung Then
 Befehlsblock
 ElseIf Bedingung Then
 Befehlsblock
 ElseIf Bedingung Then
 Befehlsblock
 ...
 Else
 Befehlsblock
 End If
%>
```



*Das letzte Else sowie der dazugehörige letzte Befehlsblock sind beide optional.*

Obiges Beispiel kann also wie folgt umgeschrieben werden:

```
<%
 Option Explicit
 Dim intTag
 intTag = Weekday(Now)
 If intTag = 1 Then
 Response.Write "Am Sonntag wird nicht gearbeitet!"
 ElseIf intTag = 2 Then
 Response.Write "Montag Morgen, da macht das Arbeiten Spaß"
 ElseIf intTag = 3 Then
 Response.Write "Dienstags ist die Montagsmüdigkeit überwunden"
 ElseIf intTag = 4 Then
 Response.Write "Am Mittwoch ist die halbe Woche schon rum"
 ElseIf intTag = 5 Then
 Response.Write "Fast geschafft, heute ist Donnerstag"
 ElseIf intTag = 6 Then
 Response.Write "Endlich Freitag, endlich Wochenende"
 Else
 Response.Write "Samstag ist Einkaufstag, nicht Arbeitstag"
 End If
%>
```

## Select Case

Schon besser, aber immer noch ein wenig zu viel Tipparbeit, zumindest nach unserem Geschmack. Aber einen Pfeil haben wir noch im Köcher: Select Case. Diese Anweisung entspricht dem switch-

Kommando von JavaScript, und die Syntax lautet wie folgt:

```
<%
 Select Case Ausdruck
 Case Werteliste
 Befehlsblock
 Case Werteliste
 Befehlsblock
 ...
 Case Else
 Befehlsblock
 End Select
%>
```



*Das Case Else sowie der darauf folgende, dazugehörige Befehlsblock sind optional. Eine Werteliste kann aus einem einzelnen Wert bestehen, oder aus mehreren, durch Kommata voneinander getrennten Werten.*

Im folgenden Code finden Sie alle Elemente von Select Case einmal im Einsatz, wieder am Beispiel der Wochentagsausgabe.

```
<%
 Option Explicit
 Dim intTag
 intTag = Weekday(Now)
 Select Case intTag
 Case 1, 7
 Response.Write "Am Wochenende wird nicht gearbeitet!"
 Case 4, 5, 6
 Response.Write "Die Woche ist schon fast herum"
 Case Else
 Response.Write "So früh in der Woche, und schon surfen?!"
 End Select
%>
```

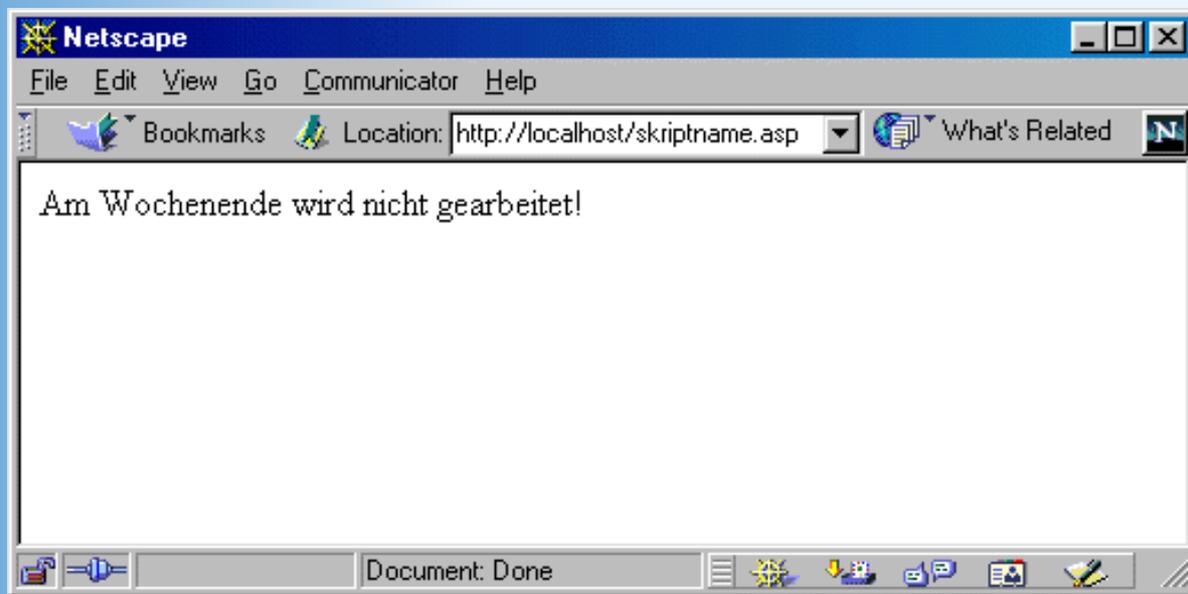


Abbildung 18.8: Die Ausgabe samstags und sonntags

## Schleifen

Die wiederholte Ausführung von Anweisungen ist ebenfalls eine häufige Anforderung an ASP-Skripte. Zu den Anwendungsmöglichkeiten gehören:

- Reine Tippersparnis - wieso mehrmals dasselbe tippen
- Verarbeitung aller Elemente einer Datengruppe (beispielsweise aller Zeichen eines Strings)
- Befristete Ausführung, bis eine Zeitspanne verstrichen oder eine andere Begebenheit sich verändert hat

## For-Schleife

Mit einer For-Schleife, der wohl einfachsten Schleifenvariante, wird eine Anweisung (oder ein Anweisungsblock) mehrfach ausgeführt. Um wieder die JavaScript-Analogie zu bemühen: auch hier gibt es eine For-Schleife, die Syntax ist bei VBScript aber etwas anders. Nachfolgend ein paar Beispiele:

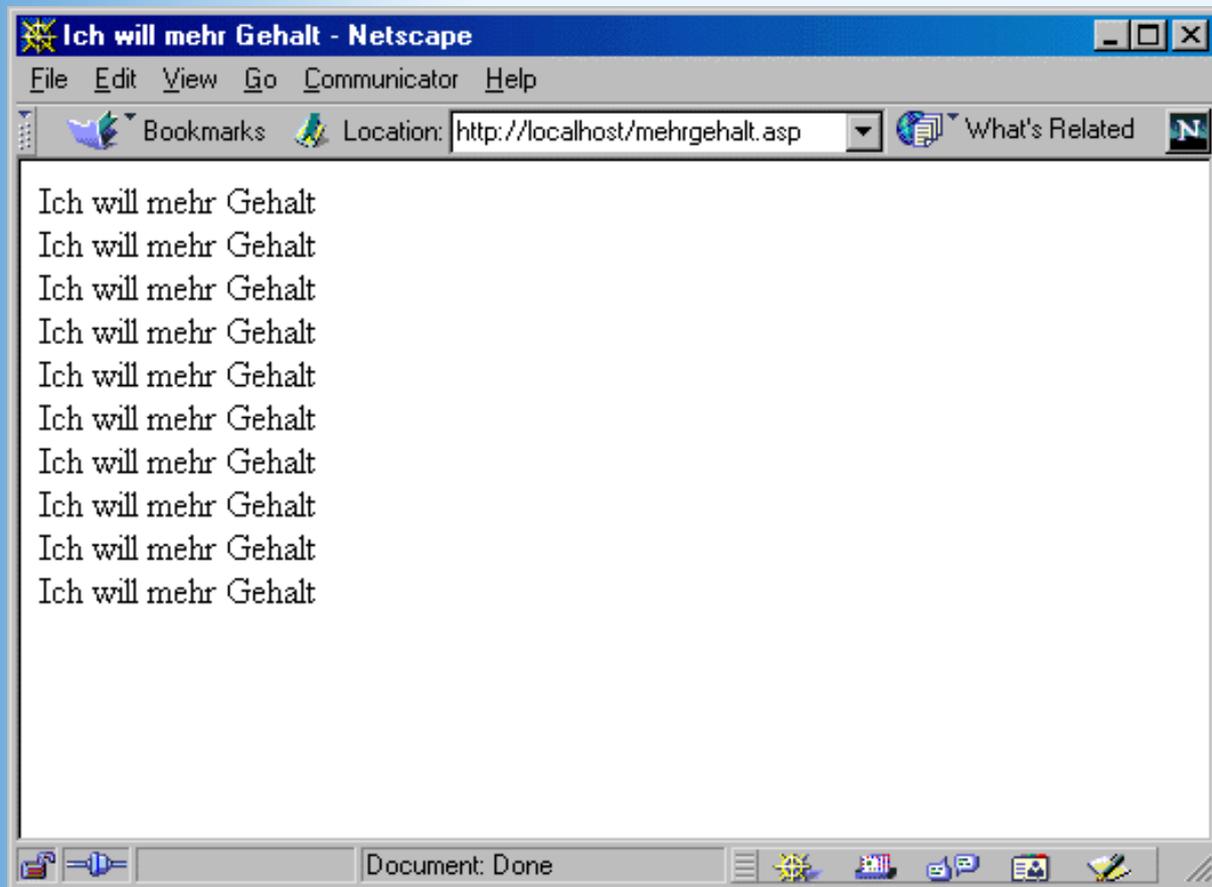
### Listing 18.6: mehrgehalt.asp

```
<%
 Option Explicit
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Ich will mehr Gehalt</title>
</head>
<body bgcolor="white">
<p>
<%
 Dim i
 For i = 1 To 10
```

```

 Response.Write "Ich will mehr Gehalt
"
Next
%>
</p>
</body>
</html>

```



**Abbildung 18.9: Gehaltsverhandlungen mit ASP**

Dieses Beispiel war noch recht einfach - zehnmal derselbe Text. Die Zählvariable (hier haben wir i gewählt) kann aber auch in den Befehlsblock eingebaut werden. Hier ein etwas komplexeres Beispiel: Eine Zeichenkette wird in seine einzelnen Zeichen zerlegt. Dazu muss die Zählvariable von 0 bis zur Länge der Zeichenkette laufen. Wie Sie aus Tabelle 18.4 ersehen können, lautet die entsprechende Zeichenkettenfunktion Len. Das entsprechende Zeichen erhalten Sie - auch aus Tabelle 18.4 ersichtlich - mit Mid. Achten Sie darauf, wie wir in der Ausgabe auf die Zählvariable zugreifen:

```

<%
 Option Explicit
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Pappenheimer</title>
</head>
<body bgcolor="white">
<p>

```

```

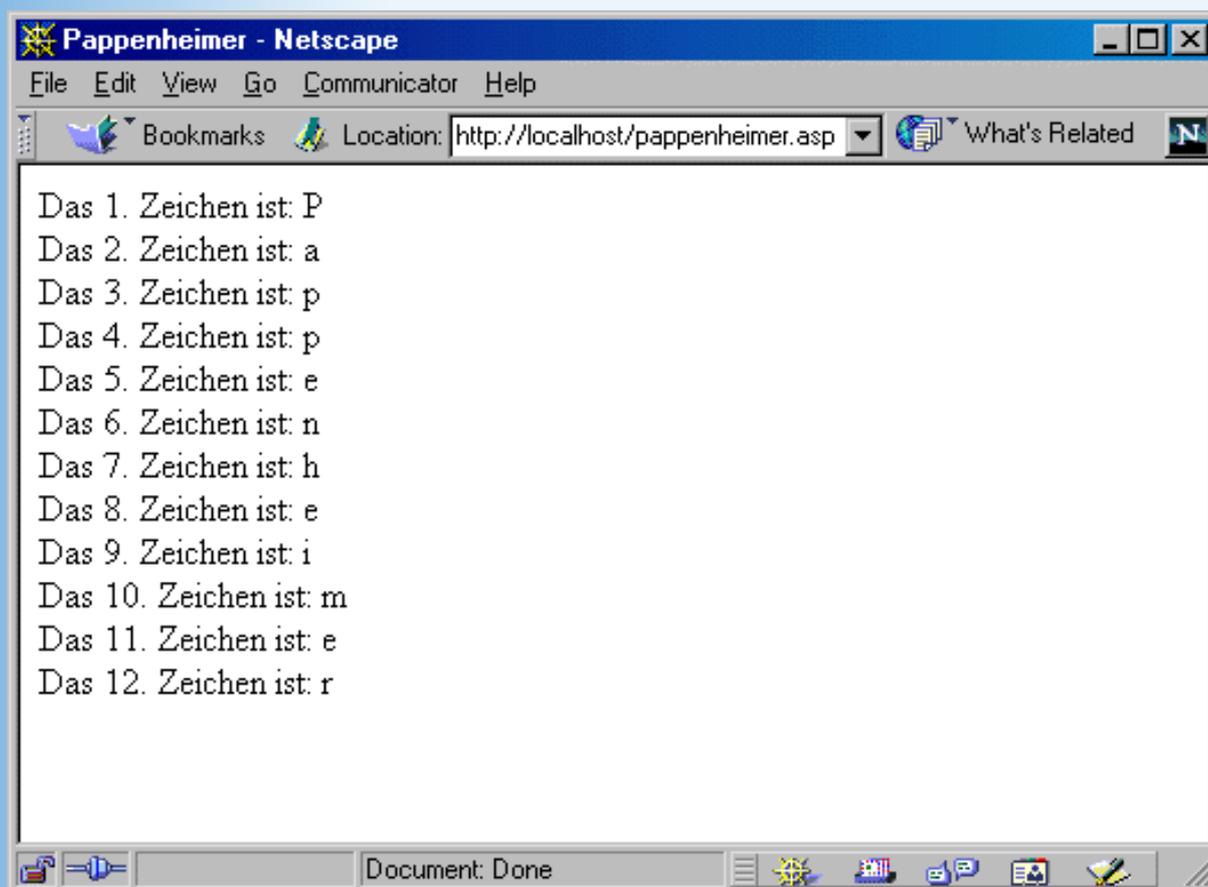
<%
 Dim i, strText
 strText = "Pappenheimer"
 For i = 1 To Len(strText)
 Response.Write "Das " & i & ". Zeichen ist: "
 Response.Write Mid(strText, i, 1)
 %>

<%
 Next
%>
</p>
</body>
</html>

```



*Sie sehen an obigem Code, wie es möglich ist, auch innerhalb einer Schleife den ASP-Code kurz zu unterbrechen und HTML-Code einzufügen.*



**Abbildung 18.10: Die einzelnen Zeichen des Wortes »Pappenheimer«**

Die Syntax der For-Schleife hat noch eine Erweiterung:

```

<%

```

```

For i = Start To Ende Step Weite
 Befehlsblock
Next
%>

```

Die Schrittweite (Weite) gibt an, um wie viel die Zählvariable pro Durchlauf erhöht wird. Der Standardwert ist 1, und da dieser fast immer verwendet wird, kann man Step Weite weglassen und die Zählvariable wird automatisch um 1 erhöht. Hin und wieder ist das Ganze aber recht nützlich, beispielsweise bei diesem (zugegebenermaßen arg konstruierten) Beispiel: Alle ungerade Zahlen von 1 bis 10 sollen ausgegeben werden:

### Listing 18.7: ungerade.asp

```

<%
 Option Explicit
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Ungerade Zahlen</title>
</head>
<body bgcolor="white">
<p>
<%
 Dim i
 For i = 1 To 10 Step 2
 Response.Write i & " "
 Next
%>
</p>
</body>
</html>

```



*Sie können eine For-Schleife durch die Anweisung Exit For vorzeitig verlassen.*

For-Schleifen sind übrigens auch im Zusammenhang mit Arrays nützlich:

### Listing 18.8: kontinente2.asp

```

<%
 Option Explicit
 Dim arystrKontinente(5), i
 arystrKontinente(0) = "Amerika"
 arystrKontinente(1) = "Afrika"

```

```
arystrKontinente(2) = "Asien"
arystrKontinente(3) = "Europa"
arystrKontinente(4) = "Antarktis"
arystrKontinente(5) = "Australien"
```

```
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Arrays und For </title>
</head>
<body bgcolor="white">
<table border="2" cellpadding="5">
<%
 For i = 0 To UBound(arystrKontinente)
%>
 <tr><td><% =arystrKontinente(i) %></td></tr>
<%
 Next
%>
</table>
</body>
</html>
```

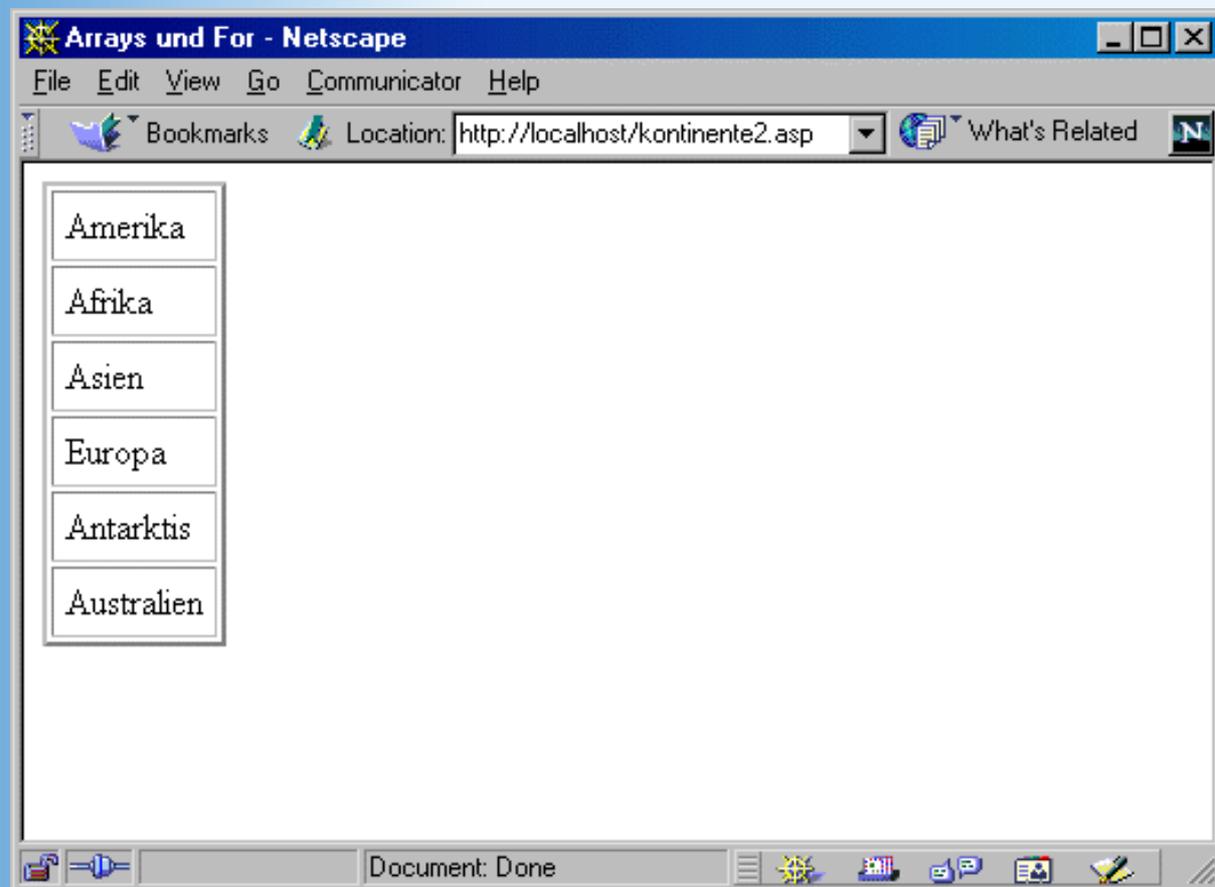


Abbildung 18.11: Die sechs Kontinente, mit einer For-Schleife ausgegeben

## Do-Loop-Schleife

Mittels Do-Loop werden Befehlsblöcke mehrfach ausgeführt, und zwar solange eine Bedingung erfüllt ist, oder bis eine Bedingung erfüllt ist. Die erste Variante (ausführen, solange eine Bedingung erfüllt ist) sieht folgendermaßen aus:

```
<%
 Do While Bedingung
 Befehlsblock
 Loop
%>
```

Die Alternative (Ausführen, bis eine Bedingung erfüllt ist):

```
<%
 Do Until Bedingung
 Befehlsblock
 Loop
%>
```

Sie können - mit ein wenig Aufwand - eine For-Schleife mit Do-Loop emulieren. Werfen Sie zunächst einen Blick auf das folgende Codestück, mit einer For-Schleife:

```
<%
 For i = 1 To 10
 Response.Write i & " "
 Next
%>
```

Das gleiche nun mit Do While-Loop:

```
<%
 i = 0
 Do While i < 10
 i = i + 1
 Response.Write i & " "
 Loop
%>
```

Die Bedingung wird also immer am Anfang des Befehlsblocks geprüft!

Und noch einmal, diesmal mit Do Until-Loop:

```
<%
 i = 0
 Do Until i = 10
 i = i + 1
 Response.Write i & " "
 Loop
%>
```

Auch hier wird wieder die Bedingung am Anfang des Befehlsblocks geprüft.

Wenn Sie eine Do-Schleife vorzeitig verlassen möchten, erreichen Sie dies mit dem Befehl Exit Do. Hier noch einmal das Beispiel von zuvor, diesmal mit Exit Do. Beachten Sie, dass die Bedingung immer erfüllt ist, und die Schleife - ohne Exit Do - endlos weiter laufen würde.

```
<%
 i = 0
 Do While i >= 0
 i = i + 1
 Response.Write i & " "
 If i = 10 Then
 Exit Do
 End If
 Loop
>%
```

## While-Wend-Schleife

Bei While-Wend handelt es sich um eine beinahe exakte Kopie von Do While-Loop. »Beinahe« deshalb, weil eine While-Schleife nicht (wie etwa Do While-Loop) vorzeitig verlassen werden kann. Ansonsten nichts Neues:

```
<%
 i = 0
 While i < 10
 i = i + 1
 Response.Write i & " "
 Wend
>%
```

## For-Each-Schleife

Die For-Each-Schleife dient dazu eine *Kollektion* zu durchlaufen. Eine Kollektion ist eine Ansammlung von Daten, und in drei Tagen (wenn wir uns den Datenbanken zuwenden) werden Sie Kollektionen genauer kennen lernen. Eine Form von Kollektionen kennen Sie bereits, und zwar Arrays.

Die Syntax von For-Each ist die folgende

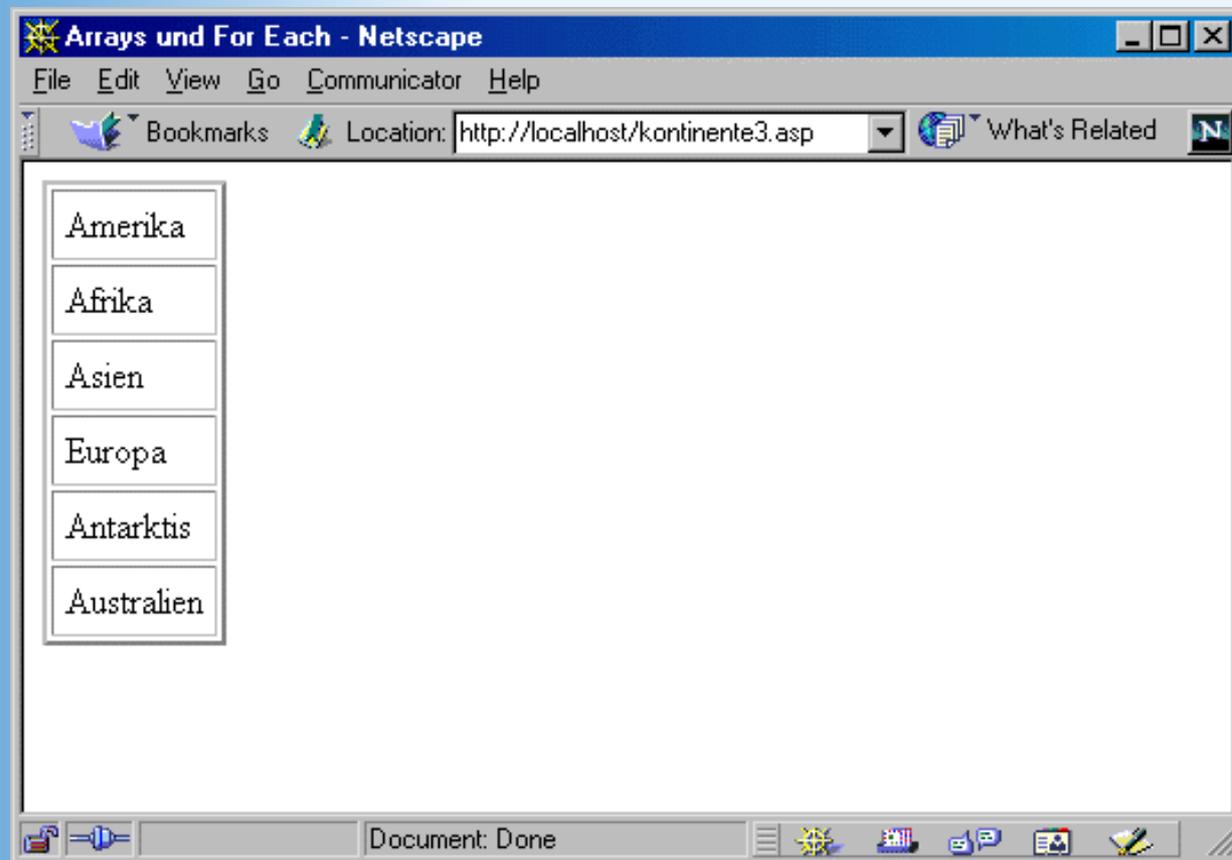
```
<%
 For Each Wert In Kollektion
 Befehlsblock
 Next
>%
```

So können Sie also - ohne sich mühsam um die Anzahl der Elemente in einem Array kümmern zu müssen - alle Elemente im Array bequem ausgeben. Alles was Sie dazu brauchen, finden Sie hier:

```
<%
 Option Explicit
 Dim arystrKontinente(5), strKontinent
```

```
arystrKontinente(0) = "Amerika"
arystrKontinente(1) = "Afrika"
arystrKontinente(2) = "Asien"
arystrKontinente(3) = "Europa"
arystrKontinente(4) = "Antarktis"
arystrKontinente(5) = "Australien"
```

```
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Arrays und For Each</title>
</head>
<body bgcolor="white">
<table border="2" cellpadding="5">
<%
 For Each strKontinent in arystrKontinente
%>
 <tr><td><% =strKontinent %></td></tr>
<%
 Next
%>
</table>
</body>
</html>
```



**Abbildung 18.12: Wieder die sechs Kontinente, diesmal mit For-Each ausgegeben**

Somit haben Sie eine erste Schnelleinführung in VBScript erhalten. Ein Blick nach draußen beschert

uns aber einsetzende Dämmerung, der Tag neigt sich aber langsam aber sicher dem Ende zu. Höchste Zeit, dass wir uns nun ein paar nützlichen Anwendungen zuwenden!

## 18.3 Datumswerte

Einer der ersten Gehversuche, sobald man über den Tellerrand von HTML schaut, besteht darin, das aktuelle Tagesdatum zu bestimmen - und auszugeben. Es ist einfach eine nette Sache, wenn man auf seiner Website das aktuelle Datum anzeigen kann, was unter anderem den netten Nebeneffekt hat, dass die Website subliminal frisch und aktuell wirkt - sie trägt ja schließlich das Tagesdatum!

Man kann natürlich das aktuelle Tagesdatum auch mit JavaScript anzeigen lassen. Dies hat aber einen kleinen Nachteil: Da JavaScript eine clientseitige Programmiersprache ist, also im Client, dem Browser, ausgeführt wird, wird auch das Datum des Clients angezeigt. Und wenn dies falsch eingestellt ist, wird auch ein falsches Datum verwendet. Bei serverseitigen Programmiersprachen wie eben ASP ist dies anders: Hier wird stets das Datum des Servers verwendet, und wenn Sie auf diesem stets die aktuelle Uhrzeit einstellen, zeigen auch Ihre Seiten immer die korrekte Zeit.

Für Datumswerte gibt es ein eigenes Datumsobjekt. Objekt deswegen, weil wir nur eine Variable benötigen, aber mittels diverser Funktionen aus dieser Variablen vielfältige Informationen ziehen können, beispielsweise den aktuellen Monat, die Uhrzeit oder sogar den Wochentag.



*Erinnern Sie sich? Wir haben heute schon einmal mit dem Wochentag gearbeitet, und zwar im Abschnitt 18.2.4!*

Ein Hinweis zur Nomenklatur: Datumsvariablen werden meist mit objDate oder objDatum oder so ähnlich bezeichnet. Das Präfix obj weist darauf hin, dass es sich um ein Objekt handelt, und der Variablenname danach sollte zumindest entfernt etwas mit dem Namensraum »Datum« zu tun haben.

Es gibt zwei Möglichkeiten, das aktuelle Tagesdatum zu erhalten: Mit Date und mit Now. Probieren wir einfach einmal beides aus:

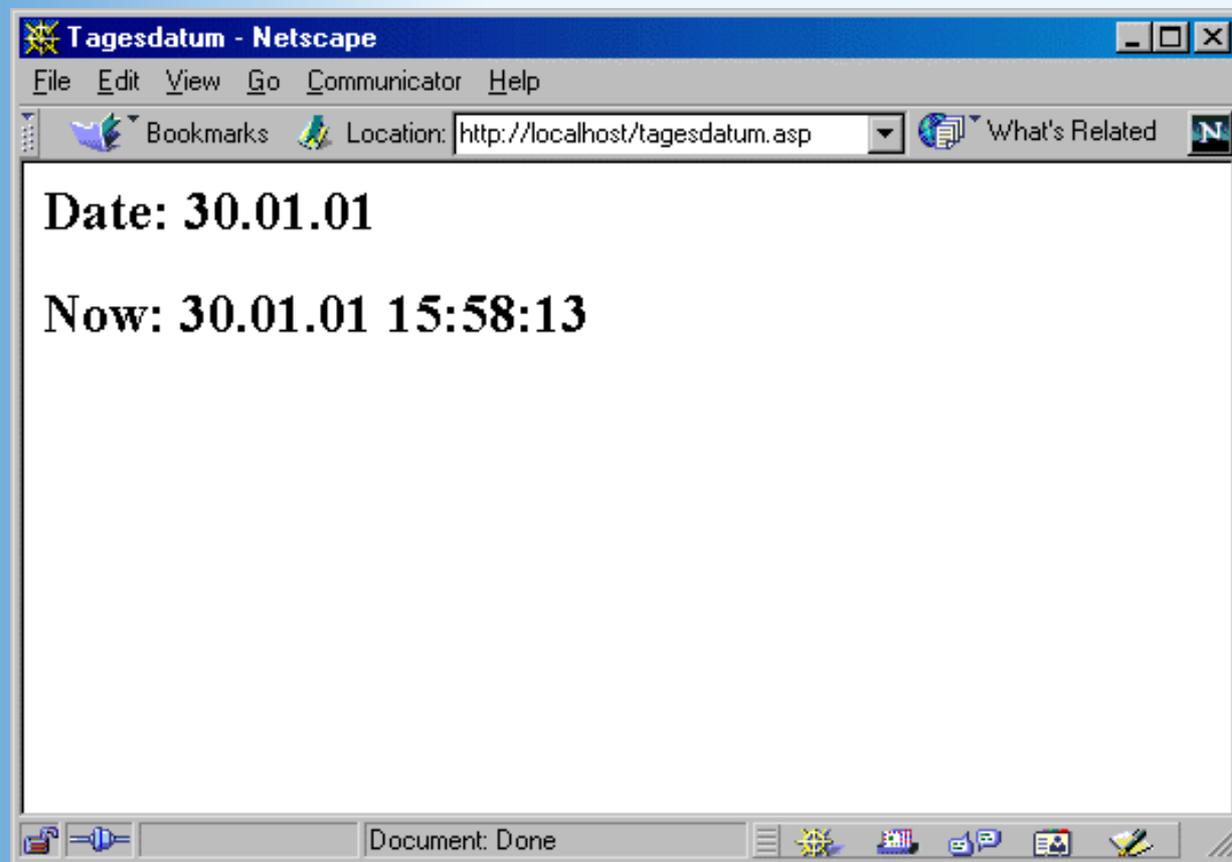
### Listing 18.9: tagesdatum.asp

```
<%
 Option Explicit
 Dim objDate, objNow
 objDate = Date
 objNow = Now
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Tagesdatum</title>
```

```

</head>
<body bgcolor="white">
<h2>Date: <% =objDate %></h2>
<h2>Now: <% =objNow %></h2>
</body>
</html>

```



**Abbildung 18.13: Der Unterschied zwischen Date und Now**

Es ist offensichtlich, was hier passiert: Date liefert das aktuelle Tagesdatum zurück, Now darüber hinaus auch noch die Uhrzeit.

Ihnen stehen nun eine Reihe von Funktionen zur Verfügung, mit denen Sie auf die einzelnen Eigenschaften des Datumsobjekts zugreifen können. In Tabelle 18.6 finden Sie eine alphabetische Übersicht.

Datumsfunktion	Beschreibung	Rückgabewert am 24.12.2001, 23:45:56 Uhr
Day()	Tag	24
Hour()	Stunde	23
Minute()	Minute	45
Month()	Monat	12
Second()	Sekunde	56
Weekday()	Wochentag (1=Sonntag, 7=Samstag)	2

Year()	Jahr	2001
--------	------	------

## Tabelle 18.6: Die Datumsfunktionen von VBScript

Sie können nun mit ein wenig Aufwand das aktuelle Tagesdatum ausgeben. Im folgenden Listing haben wir das einmal durchgeführt. Achten Sie darauf, dass wir die deutschen Monats- und Tagesnamen in Arrays speichern und dann ausgeben. Zwar stellt VBScript auch hierfür etwas zur Verfügung, aber nicht immer die deutschen Namen!

### Listing 18.10: tagesdatum\_deutsch.asp

```
<%
Option Explicit
Dim arystrMonatsnamen, arystrTagesnamen, objDate
arystrMonatsnamen = Array("", "Januar", "Februar", "Mär", _
 "April", "Mai", "Juni", "Juli", "August", "September", "Oktober", _
 "November", "Dezember")
arystrTagesnamen = Array("", "Sonntag", "Montag", "Dienstag", _
 "Mittwoch", "Donnerstag", "Freitag", "Samstag")
objDate = Now
Dim strDatum
strDatum = arystrTagesnamen(Weekday(objDate))
strDatum = strDatum & ", der "
strDatum = strDatum & Day(objDate) & ". "
strDatum = strDatum & arystrMonatsnamen(Month(objDate)) & " "
strDatum = strDatum & Year(objDate) & ", "
strDatum = strDatum & Hour(objDate) & ":"
strDatum = strDatum & Minute(objDate) & ":"
strDatum = strDatum & Second(objDate) & " Uhr."
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Tagesdatum</title>
</head>
<body bgcolor="white">
<p>Guten Tag! Heute ist <% =strDatum %></p>
</body>
</html>
```

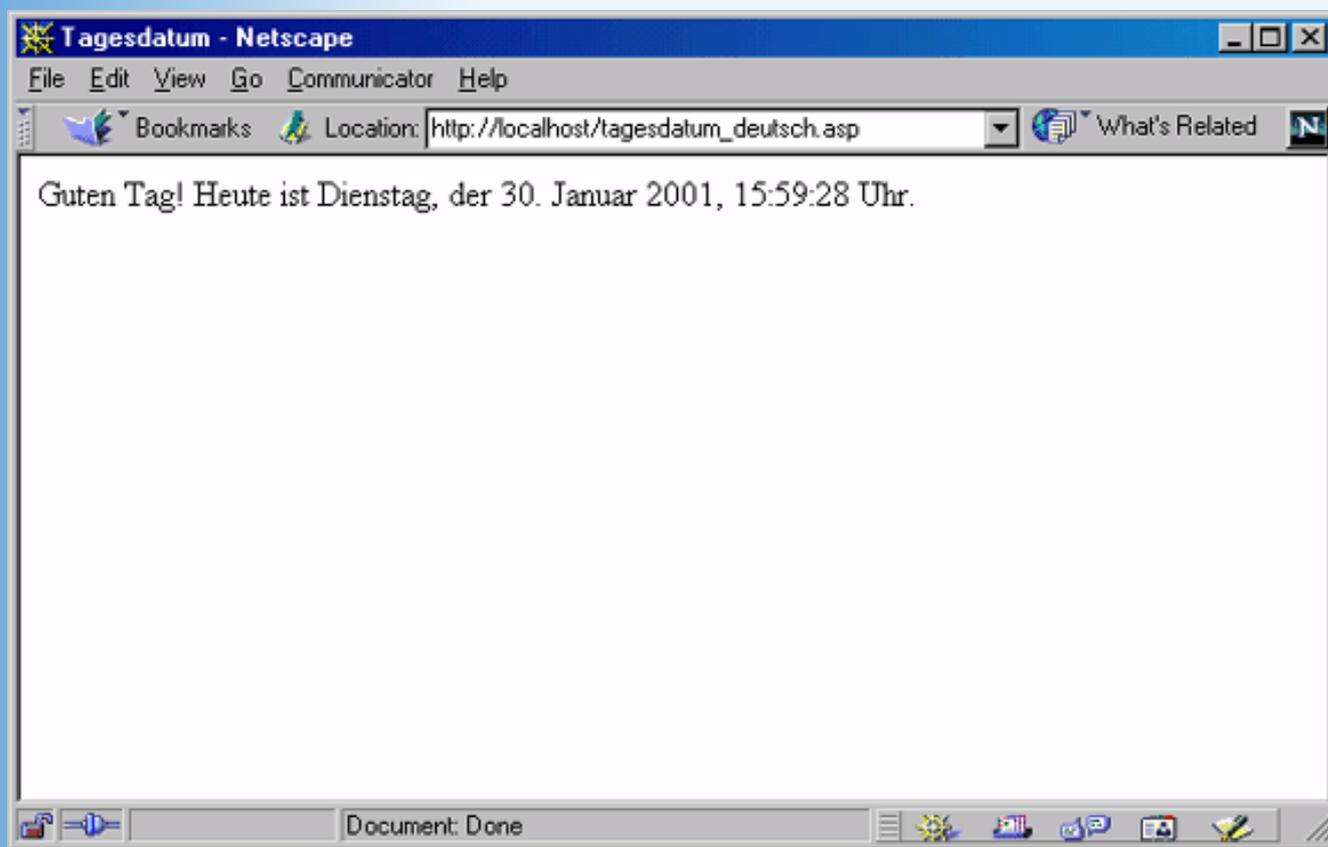


Abbildung 18.14: Datum mit ASP und VBScript



*Das jeweils erste Element jedes Arrays war ein Leerstring (»«). Der Grund: Dieses Element hat den Index 0, und es gibt weder Wochentag noch Monat 0!*

## 18.4 Auf Formulare zugreifen

Die wohl einfachste Möglichkeit, mit dem Besucher seiner Webseiten in Kontakt zu treten und Interaktion zu betreiben, liegt in der Verwendung von Formularen. In der ersten Woche haben Sie ja unter anderem eine ausführliche Einführung in die Thematik erhalten, und auch in der zweiten Woche sind Sie immer wieder auf Formularelemente gestoßen, und wie Sie mit JavaScript darauf zugreifen können. Aus diesem Grund werden wir den HTML-Anteil der folgenden Seiten relativ gering halten, damit wir uns nicht allzu oft wiederholen.

### Allgemeines

Zunächst einmal müssen Sie sich ja bei einem Formular entscheiden, ob Sie die Daten mittels POST oder mittels GET verschicken. Zur Erinnerung: Bei Versand mittels POST werden die Formulardaten als Teil des HTTP-Headers verschickt; wenn Sie GET einsetzen, werden die Daten an die URL der Seite angehängt.



*Die Länge eines URL ist von Webserver zu Webserver unterschiedlich, liegt aber in der Praxis irgendwo zwischen 500 und 2000 Zeichen. Sie sollten also umfangreiche Formulare immer via POST versenden, und nicht via GET. Ganz generell ist POST vorzuziehen.*

Zunächst zur grauen Theorie: Wenn Sie Daten via POST verschicken, können Sie mittels Request.Form darauf zugreifen. Bei GET verwenden Sie Request.QueryString.

Im folgenden Listing finden Sie das einmal umgesetzt. Sie finden dort zwei Formulare, eines verschickt die Daten mittels POST, das andere die Daten mittels GET. Das action- Attribut des Formulars ist nicht gesetzt, das heißt, die Formulardaten werden an die aktuelle Seite verschickt. Die Werte von Request.Form und Request.QueryString werden ausgegeben, und je nachdem, welches der beiden Formulare Sie verschicken, enthält einer der beiden Ausdrücke einen Wert.

### **Listing 18.11: post\_get.asp**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Formulare</title>
</head>
<body bgcolor="white">
<h2>Request.Form: <% =Request.Form %></h2>
<h2>Request.QueryString: <% =Request.QueryString %></h2>
<h3>Formular (POST)</h3>
<p>
<form method="post">
<input type="text" name="vorname1" size="20" />Vorname

<input type="text" name="nachname1" size="20" />Nachname

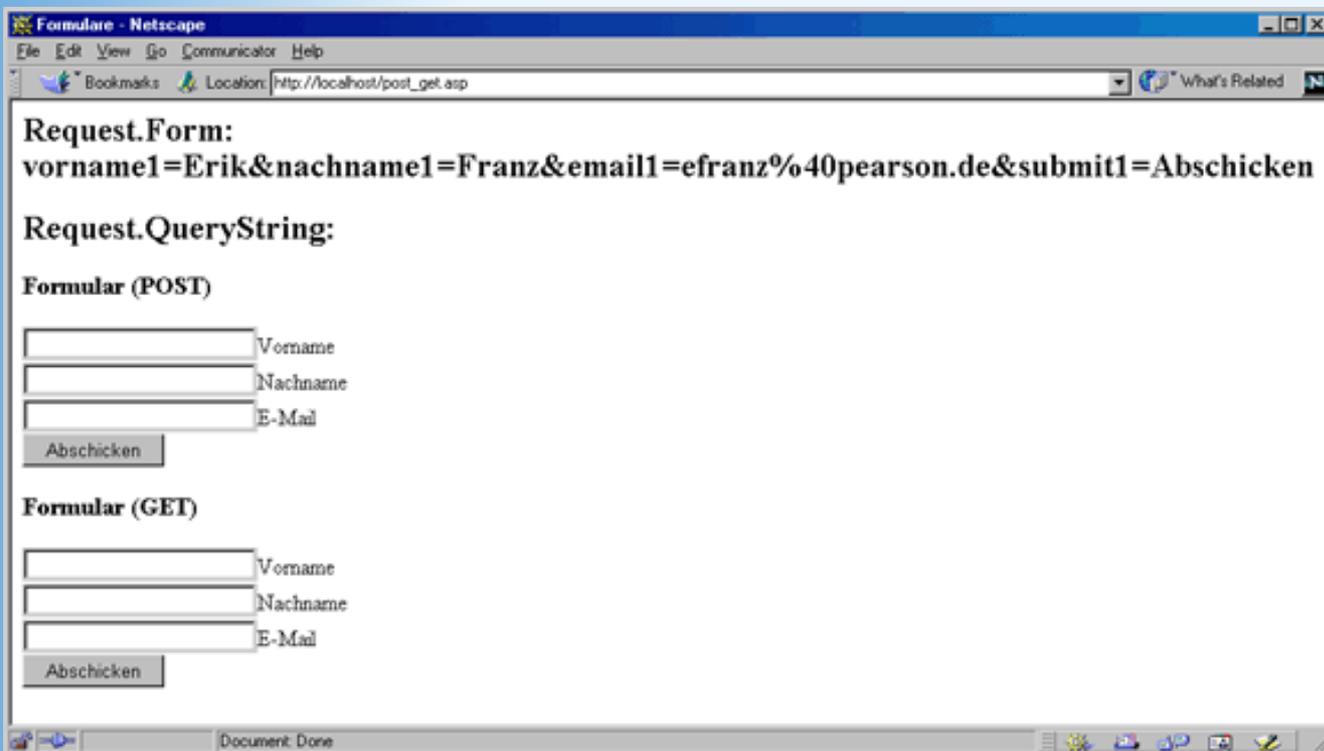
<input type="text" name="email1" size="20" />E-Mail

<input type="submit" name="submit1" value="Abschicken" />
</form>
</p>
<h3>Formular (GET)</h3>
<p>
<form method="get">
<input type="text" name="vorname2" size="20" />Vorname

<input type="text" name="nachname2" size="20" />Nachname

<input type="text" name="email2" size="20" />E-Mail

<input type="submit" name="submit2" value="Abschicken" />
</form>
</p>
</body>
</html>
```



**Abbildung 18.15: Alle Formularwerte stehen in Request.Form**

Sie sehen also: Die einzelnen name-Attribute werden durch ein Gleichheitszeichen mit den jeweiligen value-Attributen (bzw. den Eingaben in die Formularfelder) gefüllt, und die einzelnen Formularfelder durch kaufmännische Unds (&) aneinandergehängt. Sie können nun mit String-Operationen versuchen, auf die einzelnen Werte Zugriff zu erhalten, aber ASP nimmt Ihnen hier einen Großteil der Arbeit ab: Mit `Request.Form("name-Attribut")` bzw. `Request.QueryString("name-Attribut")` erhalten Sie das dazugehörige value-Attribut! Im obigen Beispiel (mitsamt Abbildung 18.15) hätte also `Request.Form("vorname1")` den Wert »Erik«.

Um das einmal auszutesten, hier noch ein kleiner Hinweis: Bei `Request.Form` bzw. `Request.QueryString` handelt es sich um eine sogenannte Kollektion. Erinnern Sie sich an den Begriff? Wir haben ihn im Zusammenhang mit den For-Each-Schleifen eingeführt. Sie können sich also mit einer For-Each-Schleife alle Elemente in der Kollektion anzeigen lassen! Das folgende Listing realisiert exakt dies; aus Platzgründen verwenden wir diesmal nur ein Formular mit POST (bei GET geht es ganz analog):

### Listing 18.12: foreach.asp

```
<%
 Option Explicit
 Dim strName
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Formulare</title>
</head>
<body bgcolor="white">
<h3>Formularwerte</h3>
```

```

<p>
<%
 For Each strName in Request.Form
 Response.Write "" & strName & ": "
 Response.Write Request.Form(strName) & "
"
 Next
%>
</p>
<h3>Formular (POST)</h3>
<p>
<form method="post">
<input type="text" name="vorname" size="20" />Vorname

<input type="text" name="nachname" size="20" />Nachname

<input type="text" name="email" size="20" />E-Mail

<input type="submit" name="submit" value="Abschicken" />
</form>
</p>
</body>
</html>

```

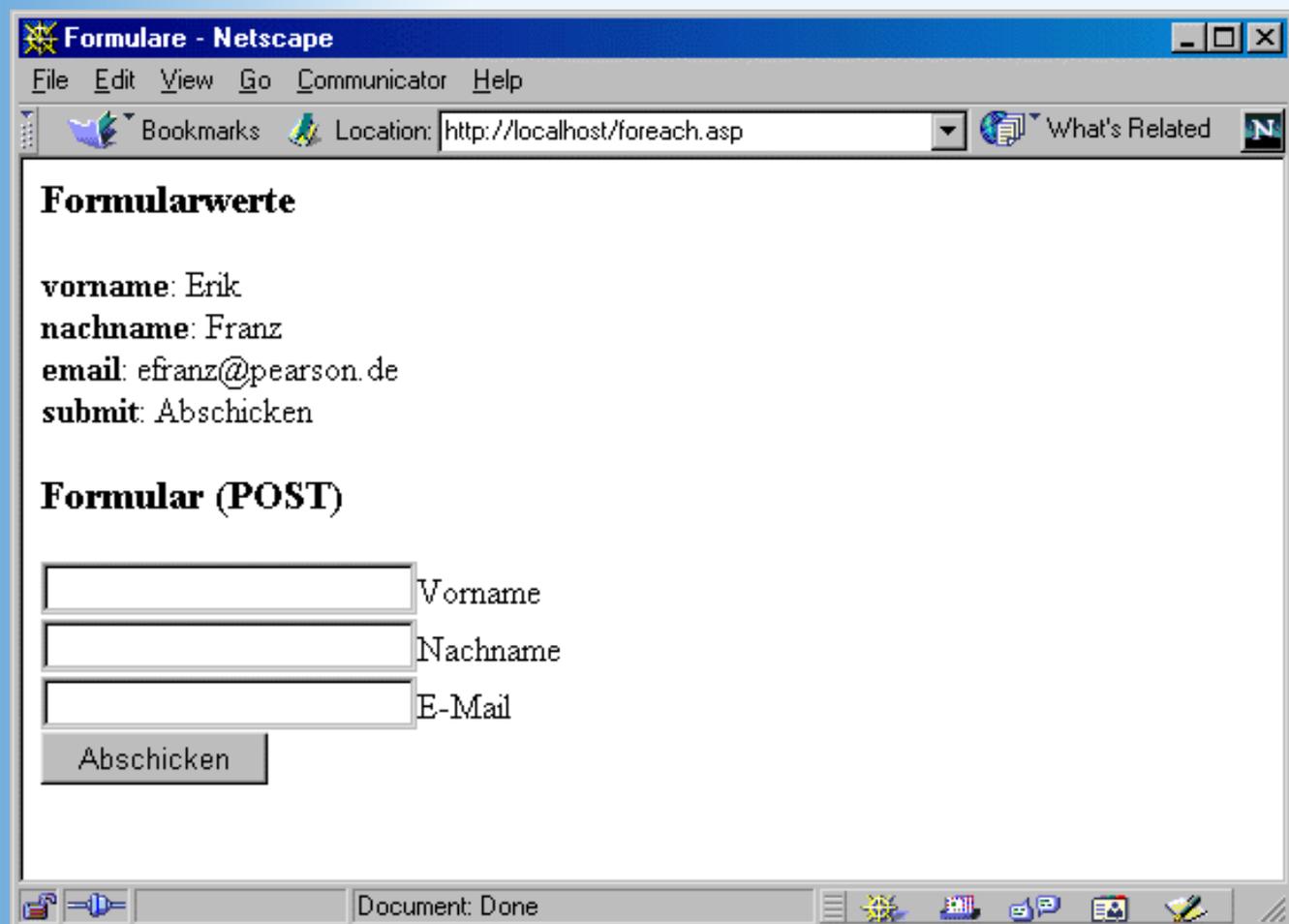


Abbildung 18.16: Die Formularangaben werden einzeln mit For-Each ausgegeben



*Ein Tipp für ganz Schreibfaule: Sie können auch direkt über Request("name-Attribut") auf die einzelnen value-Attribute zugreifen. Wenn Sie nämlich nur Request angeben, wird zunächst Request.Form durchsucht, und dann Request.QueryString. In diesem Fall ist es also völlig egal, ob das Formular mittels POST oder mittels GET verschickt worden ist. Ob das dann aber noch sauberer Programmierstil ist, sei einmal dahingestellt.*

## Unterschiede bei den einzelnen Formularfeldern

Oben aufgeführte Regel gilt tatsächlich in den meisten Fällen: Geben Sie das name-Attribut an, und Sie erhalten das dazugehörige value-Attribut. Bei einigen der Formularelemente gibt es jedoch kleine, aber feine Unterschiede. Werfen wir also einen genaueren Blick auf die verschiedenen Möglichkeiten, die einem HTML und ASP bieten.

### Textfelder / Passwortfelder

Alles, was durch die HTML-Konstrukte `<input type="text">`, `<input type="password">` und `<textarea>` angegeben wird, folgt von ASP-Seiten aus einer Regel: `Request.Form("name-Attribut")` bzw. `Request.QueryString("name-Attribut")` liefert das dazugehörige value-Attribut zurück. Ein möglicher Code zur Ausgabe kann dann folgendermaßen aussehen:

```
Sie haben als E-Mail-Adresse
<% =Request("email") %>
angegeben!
```

### Auswahllisten

Noch einmal zur Erinnerung: Eine Auswahlliste ist wie folgt aufgebaut:

```
<select name="Name">
 <option value="Wert1">Beschriftung 1</option>
 <option value="Wert2">Beschriftung 2</option>
 <!-- ... -->
</select>
```

Wenn Sie obigen HTML-Code als Teil eines Formulars an ein ASP-Skript verschicken, erhalten Sie dort dann mittels `Request("Name")` das value-Attribut der ausgewählten Option. Etwas anderes ist es jedoch, wenn Sie `<select multiple>` verwenden. Dann stehen in `Request("Name")` die value-Attribute *aller* ausgewählten Elemente, durch Kommata voneinander getrennt. Auch hier können Sie beispielsweise mit `InStr` und Konsorten die einzelnen Werte herausbekommen, aber der alte For-Each-Trick funktioniert ebenfalls tadellos: Setzen Sie diese Schleife ein, und Sie erhalten alle einzelnen Werte. Folgender Beispielcode funktioniert prinzipiell sowohl bei `<select>` als auch bei `<select multiple>`:

```
<p>Sie haben folgende Werte ausgewählt:</p>
<p>
<%
 Dim strWert
 For Each strWert in Request("Name")
 Response.Write strWert & "
"
 Next
```

```
%>
</p>
```

## Checkboxes

Auch, wenn Checkboxes in der Regel in Rudeln auftreten (will sagen: eine Checkbox kommt selten allein), so werden dennoch Checkboxes jeweils einzeln und unabhängig voneinander behandelt. Aus diesem Grund ist der Zugriff auch hier recht einfach. Ist die Checkbox angekreuzt, so enthält Request("name-Attribut") das value-Attribut der Checkbox, ansonsten ist Request("name-Attribut") leer.



*Wenn Sie kein value-Attribut angeben, übergeben die meisten Browser als value-Attribut den String »on«. Schlussfolgerung: Geben Sie möglichst immer ein value-Attribut an!*

Der folgende Code zeigt exemplarisch, wie Sie feststellen können, ob eine Checkbox angekreuzt ist oder nicht:

```
<%
 If Request.Form("Name")<>" " Then
 Response.Write "Checkbox angekreuzt!"
 Else
 Response.Write "Checkbox nicht angekreuzt!"
 End If
%>
```

## Radiobuttons

Im Gegensatz zu Checkboxes werden Radiobuttons gruppenweise verwaltet: Von allen Radiobuttons innerhalb einer Gruppe darf immer nur höchstens einer ausgewählt werden. Damit ist ziemlich klar, was Request("name-Attribut") zurückliefert: Das value-Attribut des ausgewählten Radio-Buttons:

Sie haben den Radiobutton `<% =Request("Name") %>` ausgewählt!

## Datei-Uploads

Die Möglichkeit, Dateien auf den Server zu überspielen, ist eine gute Sache - aber nicht, wenn Sie ASP einsetzen. Mit den Standardmitteln von ASP ist es sehr schwierig, auf die übermittelten Dateien zuzugreifen. Am einfachsten ist es hier, Sie verwenden ein zusätzliches Programm, oder Sie schwenken zu PHP über. Aus diesem Grund behandeln wir dieses Thema heute nicht, aber morgen, wenn Sie die Open-Source-Konkurrenz zu ASP kennen lernen, erfahren Sie mehr darüber.

## Versende-Schaltflächen

Mit `<input type="submit">` wird ein Formular verschickt. Und auch hier können Sie mit Request("name-Attribut") auf das dazugehörige value-Attribut (was in diesem Falle gleich der Beschriftung der

Schaltfläche ist) zugreifen. Auf den ersten Blick ist das nicht sinnvoll, aber auf den zweiten Blick ist das sehr nützlich. Stellen Sie sich vor, Sie verschicken ein Formular auf dieselbe ASP-Seite, und wollen dieses Formular dann verarbeiten. Dazu müssen Sie zunächst feststellen, ob das Formular verschickt worden ist (dann müssen die Daten verarbeitet werden), oder ob diese Seite direkt aufgerufen worden ist (dann muss das Formular angezeigt werden). Hier ein exemplarischer Code für diese Aufgabe:

```
<%
 If Request.Form("Versendebutton") = "Abschicken" Then
 ' Formulardaten verarbeiten
 Else
 ' Formular anzeigen
 End If
>%
```

## Praxisbeispiele

Sie wissen nun bereits schon fast alles, was man zum Umgang mit Formularen können muss. Wir wollen Sie aber nicht aus diesem Abschnitt entlassen, ohne vorher zwei Praxisbeispiele angeboten zu haben. Viel Spaß beim Ausprobieren und Erweitern!

### Anzeige der Daten

Im ersten Beispiel werden alle angegebenen Daten ausgegeben. Wir greifen dazu auf jedes Formularfeld einzeln zu, und bedienen uns nicht der (faulen) For-Each-Schleife. In der Praxis könnte man dieses Beispiel etwa dahingehend erweitern, dass die Daten parallel dazu noch in eine Datenbank geschrieben werden (in drei Tagen mehr dazu!). Von der Programmierung her bringt der Code wenig Neues, so dass wir ihn hier ohne weitere Umstände direkt wiedergeben:

#### Listing 18.13: daten\_anzeigen.asp

```
<%
 Option Explicit
 Dim strAnrede, strOS, i
>%
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Formulare</title>
</head>
<body bgcolor="white">
<%
 If Request.Form("submit")="Abschicken" Then
>%
<h3>Formularwerte</h3>
<p>
<%
 Select Case Request.Form("Geschlecht")
 Case "m"
```

```

 strAnrede = "Herr"
 Case "f"
 strAnrede = "Frau"
 Case Else
 strAnrede = "Herr/Frau"
End Select
Response.Write "Hallo " & strAnrede & " "
Response.Write Request.Form("Vorname") & " "
Response.Write Request.Form("Nachname") & "!
"
%>
 Sie setzen die folgenden Betriebssysteme ein:
<%
 If Request.Form("Windows")<>" " Then
 Response.Write "Windows "
 End If
 If Request.Form("Linux")<>" " Then
 Response.Write "Linux "
 End If
%>

Schön, dass Ihnen Tag <% =Request.Form("Tag") %>
 so gut gefällt.
</p>
<%
 Else
%>
<h3>Formular (POST)</h3>
<p>
<form method="post">
<input type="text" name="Vorname" size="20" />Vorname

<input type="text" name="Nachname" size="20" />Nachname

<input type="radio" name="Geschlecht" value="m">männlich
<input type="radio" name="Geschlecht" value="f">weiblich

Ich nutze
<input type="checkbox" name="Windows" value="ja">Windows
<input type="checkbox" name="Linux" value="ja">Linux

<select name="Tag" size="1">
<%
 For i=1 To 21
%>
<option value="<% =i %>"><% =i %></option>
<%
 Next
%>
</select>Dieser Tag gefällt mir am Besten

<input type="submit" name="submit" value="Abschicken" />
</form>
</p>
<%
 End If
%>
</body>

```

</html>



Ein kleiner Kniff verdient noch der Erwähnung. Für die Auswahlliste mit den 21 Tagen wäre normalerweise recht viel HTML-Code nötig gewesen. Mit einer kleinen For-Schleife geht das jedoch in wenigen Zeilen.

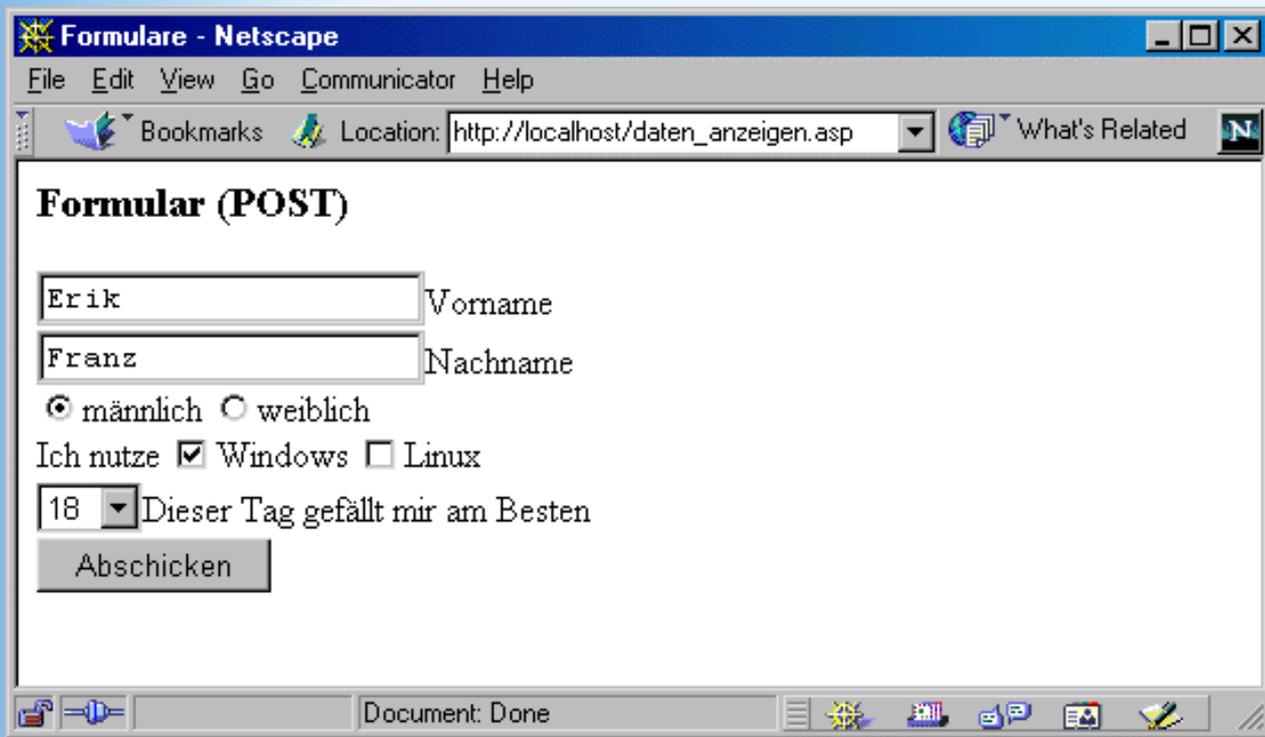
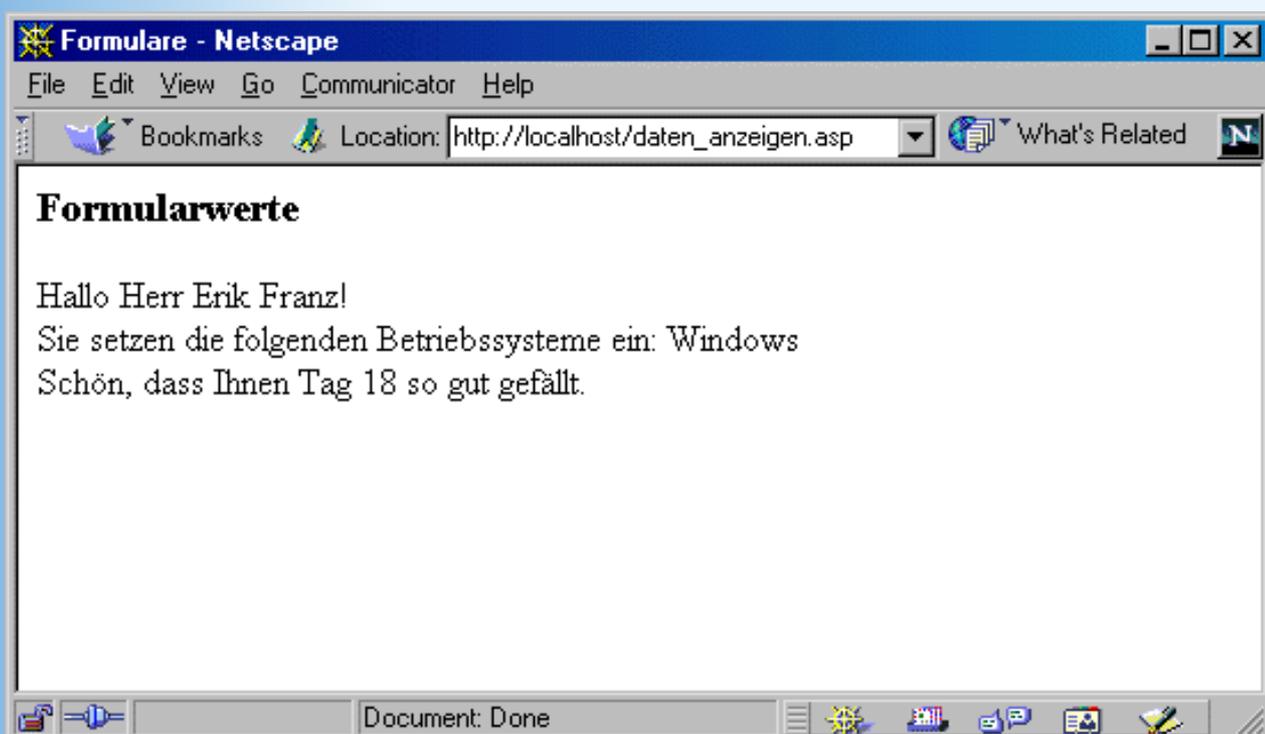


Abbildung 18.17: Das Formular ...



## Vollständigkeitsüberprüfung

Websurfer füllen ungern Formulare aus, denn das dauert Zeit, und man muss doch immer mal wieder einige persönliche Daten preisgeben. Wir möchten uns nicht in die Diskussion um den Datenschutz einmischen, aber es ist klar, dass auf der anderen Seite der Betreiber einer Website Interesse daran hat, dass das Formular auch vollständig ausgefüllt wird. Mit JavaScript kann man ein Formular auf Vollständigkeit überprüfen, aber JavaScript lässt sich ausschalten. Sie sollten deswegen auch auf der Serverseite eine Vollständigkeitsüberprüfung vornehmen. Im folgenden Listing wird das so realisiert: Nach dem Abschicken des Formulars wird überprüft, ob alle Formularfelder ausgefüllt worden sind. Falls ja, wird ein Dankestext angezeigt, falls nein, wird das Formular nochmals ausgegeben.

### Listing 18.14: vollstaendig.asp

```
<%
Option Explicit
Dim bVollstaendig, i
bVollstaendig = False
If Request.Form("submit")="Abschicken" Then
 bVollstaendig = True
 If Request.Form("Geschlecht") = "" Then
 bVollstaendig = False
 End If
 If Request.Form("Vorname") = "" Then
 bVollstaendig = False
 End If
 If Request.Form("Nachname") = "" Then
 bVollstaendig = False
 End If
 If Request.Form("Windows") = "" And Request.Form("Linux") = "" Then
 bVollstaendig = False
 End If
End If
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Formulare</title>
</head>
<body bgcolor="white">
<h3>Formularversand</h3>
<p>
<%
 If bVollstaendig Then
 Response.Write "Danke für Ihre Angaben!"
 Else
%>
<form method="post">
<input type="text" name="Vorname" size="20" />Vorname

```

```

<input type="text" name="Nachname" size="20" />Nachname

<input type="radio" name="Geschlecht" value="m">mänlich
<input type="radio" name="Geschlecht" value="f">weiblich

Ich nutze
<input type="checkbox" name="Windows" value="ja">Windows
<input type="checkbox" name="Linux" value="ja">Linux

<select name="Tag" size="1">
<%
 For i=1 To 21
%>
<option value="<% =i %>"><% =i %></option>
<%
 Next
%>
</select>Dieser Tag gefällt mir am Besten

<input type="submit" name="submit" value="Abschicken" />
</form>
</p>
<%
 End If
%>
</p>
</body>
</html>

```

In der booleschen Variablen `bVollstaendig` steht, ob das Formular vollständig ausgefüllt worden ist. Standardwert ist `False`, das heißt das Formular muss (ggf. erneut) angezeigt werden. Beim Formularversand wird der Wert der Variablen auf `True` gesetzt und dann die Felder einzeln auf Vollständigkeit überprüft. Wurde ein Feld nicht ausgefüllt, wird die Variable wieder auf `False` zurückgesetzt. Im `<body>`-Abschnitt dann wird der Wert der Variablen überprüft: Bei `False` wird das Formular angezeigt, bei `True` ein Dankestext.

Eine weitere Möglichkeit besteht darin, den Benutzer auf eine Dankesseite weiterzuleiten, wenn das Formular korrekt ausgefüllt worden ist. Dazu dient die Anweisung `Response.Redirect`. Als Parameter wird dazu die URL übergeben, auf die weitergeleitet werden soll.



*Diese Anweisung muss vor dem ersten HTML-Code ausgeführt werden. Oder, mit anderen Worten: Sobald Sie HTML-Ausgaben haben, können Sie `Response.Redirect` nicht mehr einsetzen.*

Im obigen Beispiel lässt sich die Weiterleitung schnell einbauen. Aus Platzgründen wiederholen wir hier nicht das komplette, doch ziemlich lange Listing, sondern nur das Codestück vor `<!DOCTYPE:`

**Listing 18.15: weiterleitung.asp (Ausschnitt)**

```

<%
Option Explicit
Dim bVollstaendig, i
bVollstaendig = False
If Request.Form("submit")="Abschicken" Then
 bVollstaendig = True
 If Request.Form("Geschlecht") = "" Then
 bVollstaendig = False
 End If
 If Request.Form("Vorname") = "" Then
 bVollstaendig = False
 End If
 If Request.Form("Nachname") = "" Then
 bVollstaendig = False
 End If
 If Request.Form("Windows") = "" And Request.Form("Linux") = "" Then
 bVollstaendig = False
 End If
End If
If bVollstaendig = True Then
 Response.Redirect "danke.asp"
End If
%>

```

Auf der Seite *danke.asp* (im selben Verzeichnis wie das Formular selbst) steht dann ein geeigneter Dankertext.

## 18.5 Mit Dateien arbeiten

Das Thema Dateihandling ist auch in ASP vorgesehen worden. Auch hier gibt es mannigfaltige Einsatzmöglichkeiten, wir beschränken uns aber hier aus Platz- und Zeitgründen (die Sonne dürfte nun auch bei Ihnen schon fast untergegangen sein) auf das Schreiben in eine Datei. Um dies zu bewerkstelligen, müssen Sie den folgenden Code verwenden:

```

<%
Dim objFSO, objFile
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.OpenTextFile("formular.txt", 8, True)
%>

```

Damit öffnen Sie eine Verbindung zur Datei *formular.txt*. Der Wert 8 bedeutet, dass an diese Datei Daten angehängt werden (zum Überschreiben müssten Sie hier eine 2 angeben). Nun können Sie mit `objFile.Write "Text"` einen Text in die Datei schreiben; `objFile.WriteLine "Text"` schreibt außer dem String auch noch einen Zeilensprung in die Datei.

Am Ende der Schreiboperationen können Sie die Datei wieder schließen, und im System belegte Ressourcen wieder freigeben. Das geht folgendermaßen:

```

<%
objFile.Close 'Schließt Datei

```

```
Set objFile = Nothing 'Löscht Objekt aus dem Speicher
Set objFSO = Nothing 'Löscht Objekt aus dem Speicher
%>
```

Das war auch schon alles - zumindest alles, was fürs Schreiben in eine Datei interessant ist. Viele Hosters bieten keinen Datenbankzugriff in Ihren Paketen an, Schreibzugriffe sind aber oftmals möglich. Aus diesem Grund kann es sich lohnen, Formulareingaben in einer Textdatei zu speichern. Im folgenden Beispiel wollen wir dies einmal mit dem altbekannten Formular durchführen. Die Formularangaben werden in eine Textdatei geschrieben, und dabei durch Tabulatorzeichen voneinander getrennt. Dies hat den Vorteil, dass die Textdatei später einmal leicht in eine Tabellenkalkulation wie beispielsweise Excel oder auch in eine Datenbank importiert werden kann.

Es fehlt Ihnen nur noch ein Detail, um das durchführen zu können. Wie erzeugen Sie ein Tabulatorsymbol? Die Antwort: Mit vbTab.

Im Folgenden nun schon das Listing, das die Daten in die Textdatei schreibt. Die Vollständigkeitsüberprüfung haben wir hier aus Gründen der Einfachheit weggelassen, Sie können sie aber wie oben gezeigt implementieren.

### Listing 18.16: dateizugriff.asp

```
<%
Option Explicit
Dim i
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Formulare</title>
</head>
<body bgcolor="white">
<%
If Request.Form("submit")="Abschicken" Then
Dim objFSO, objFile
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.OpenTextFile("formular.txt", 8, True)
objFile.Write Request.Form("Vorname") & vbTab
objFile.Write Request.Form("Nachname") & vbTab
objFile.Write Request.Form("Geschlecht") & vbTab
objFile.Write Request.Form("Windows") & vbTab
objFile.Write Request.Form("Linux") & vbTab
objFile.WriteLine Request.Form("Tag")
objFile.Close
Set objFile = Nothing
Set objFSO = Nothing
Response.Write "<p>Vielen Dank für Ihre Angaben!</p>"
Else
%>
<h3>Formular (POST)</h3>
<p>
```

```

<form method="post">
<input type="text" name="Vorname" size="20" />Vorname

<input type="text" name="Nachname" size="20" />Nachname

<input type="radio" name="Geschlecht" value="m">männlich
<input type="radio" name="Geschlecht" value="f">weiblich

Ich nutze
<input type="checkbox" name="Windows" value="ja">Windows
<input type="checkbox" name="Linux" value="ja">Linux

<select name="Tag" size="1">
<%
 For i=1 To 21
%>
<option value="<% =i %>"><% =i %></option>
<%
 Next
%>
</select>Dieser Tag gefällt mir am Besten

<input type="submit" name="submit" value="Abschicken" />
</form>
</p>
<%
 End If
%>
</body>
</html>

```



*Der Speicherort der Datei formular.txt ist je nach System unterschiedlich, meistens befindet sich die Datei im Hauptverzeichnis der Festplatte C:. Sie können jedoch einen absoluten Pfad zur Datei angeben und diese somit an einem beliebigen Ort ablegen. Achten Sie aber unbedingt darauf, dass die Datei nicht via HTTP (also über Eingabe einer URL) heruntergeladen werden kann, das ist ein großes Sicherheitsloch (jeder könnte dann mit ein bisschen Raten auf die Datei zugreifen und dann alle Formulareinträge lesen).*



## 18.6 Cookies

HTTP, das HyperText Transfer Protocol, das für den Datenfluss im World Wide Web zuständig ist, arbeitet nach folgendem Prinzip: Verbindung aufbauen - Daten schicken - Verbindung schließen. Wenn Sie also eine halbe Stunde lang auf der Website des Markt+Technik Verlags ([www.mut.de](http://www.mut.de)) herumsurfen, dann wird eine Vielzahl von Verbindungen zum Webserver geöffnet und auch wieder geschlossen. Der Webserver jedoch weiß eigentlich nicht, dass das genau eine Person war, die permanent Daten von ihm angefordert hat. Aus diesem Grund hat Netscape, die Firma, die auch den gleichnamigen Browser entwickelt hat, das Konzept der Cookies ersonnen. Cookies sind kleine Textinformationen, die auf der Festplatte des Browsers abgelegt werden und bei jeder neuen Anforderung an den Webserver wieder mitgeschickt werden.



*Wir wollen uns auch hier aus den Diskussionen über Sinn und Unsinn von Cookies heraushalten, aber wir vertreten die Meinung, dass Cookies (richtig eingesetzt) nicht schädlich sind, und dass es bei manchen Applikationen zu Cookies auch keine wirklichen Alternativen gibt - insbesondere in Verbindung mit ASP.*

### Was sind Cookies?

Um das Konzept der Cookies noch einmal zu verdeutlichen: Stellen Sie sich vor, Ihr Name ist in einem Cookie gespeichert. Bei jeder Anfrage an den Webserver wird diese Information mitgeschickt. Auf der Serverseite kann nun beispielsweise ein ASP-Skript diesen Namen auslesen und auf jeder Seite ausgeben, Sie haben also eine einfache Form der Personalisierung.

Bei der Verwendung von Cookies gibt es aber eine Reihe von Einschränkungen:

- Cookies werden nur an den Webserver zurückgeschickt, von dem sie stammen. Wenn Sie also auf der Website von Markt+Technik einen Cookie erhalten, können den alle Konkurrenten nicht lesen (aber dort sollten Sie eh nicht herumsurfen).
- Der Browser speichert insgesamt nur eine begrenzte Zahl von Cookies, in der Regel 300 Stück. Ist der Cookiespeicher voll, und neue Cookies treffen beim Browser ein, werden *irgendwelche* alten Cookies gelöscht.
- Pro Domain speichert der Browser nur eine begrenzte Zahl von Cookies, in der Regel 20 Stück. Auch hier gilt: Wenn die Grenze erreicht ist, werden alte Cookies gelöscht.
- Die Daten innerhalb eines Cookies dürfen eine bestimmte Obergrenze nicht überschreiten, in der Regel 4 Kbyte.
- Cookies werden nicht browserübergreifend gespeichert, will sagen: Der Internet Explorer kann nicht auf Cookies des Netscape Navigator zugreifen und umgekehrt.

Die offizielle Cookie-Spezifikation finden Sie im Internet unter [http://www.netscape.com/newsref/std/cookie\\_spec.html](http://www.netscape.com/newsref/std/cookie_spec.html). Die dortigen Angaben haben jedoch Empfehlungscharakter, und es liegt in den Händen der Browserhersteller, wie sie die dortigen Vorgaben umsetzen. Aus diesem Grund haben wir die Formulierungen oben, insbesondere in Bezug auf die Obergrenzen, absichtlich

unscharf gelassen.

## Cookies mit ASP

Wie Sie gesehen haben, können Sie auch mit JavaScript Cookies setzen. Dies ist aber nur eine Art Entgegenkommen der Browserhersteller, denn normalerweise gelten die folgenden Maxime:

- Wenn ein Cookie beim Client gesetzt werden soll, so wird es vom Webserver als Teil des HTTP-Headers mitgeschickt.
- Wenn ein Cookie vom Server gelesen werden soll, dann wird die Information aus dem HTTP-Header einer Client-Anforderung ausgelesen.

Sie müssen also unterscheiden, ob Sie Cookies setzen oder lesen möchten.

## Cookies setzen

Wie bereits erwähnt, werden Cookies vom Server als Teil der HTTP-Header-Informationen gesetzt. Von ASP-Seiten aus steht Ihnen hierzu die Kollektion `Response.Cookies` zur Verfügung. Sie können folgendermaßen ein Cookie setzen:

```
<%
 Response.Cookies("Name") = "Erik"
%>
```



*Dieser Code muss vor jeder HTML-Ausgabe stehen, damit der ASP-Interpreter die Cookie-Informationen in den HTTP-Header schreiben kann!*

Hier ein etwas längeres Beispiel: Der Benutzer gibt seine Lieblingsfarbe in ein Texteingabefeld ein; sie wird dann als Cookie gespeichert:

### Listing 18.17: cookie\_schreiben1.asp

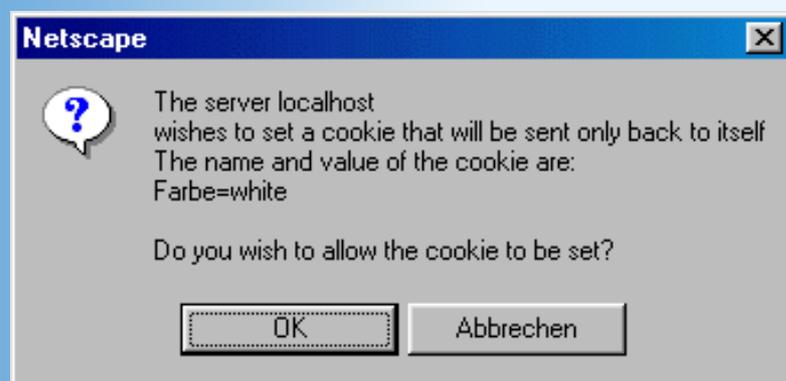
```
<%
 If Request.Form("submit") = "Abschicken" Then
 Response.Cookies("Farbe") = Request.Form("Farbe")
 End If
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Cookies</title>
</head>
<body bgcolor="white">
<h2>Cookies</h2>
```

```

<%
 If Request.Form("submit") = "Abschicken" Then
%>
Sie haben <%=Request.Form("Farbe") %>
 als Ihre Lieblingsfarbe angegeben!
<%
 Else
%>
<form method="post">
<input type="text" name="Farbe" size="20" /> Lieblingsfarbe

<input type="submit" name="submit" value="Abschicken">
</form>
<%
 End If
%>
</body>
</html>

```



**Abbildung 18.20: Der Browser will einen Cookie setzen**

## Cookies dauerhaft speichern

Wenn Sie, wie zuvor beschrieben, einen Cookie setzen, dann wird der wieder automatisch gelöscht, wenn Sie den Browser beenden. Man spricht in diesem Zusammenhang auch von *temporären Cookies*. Interessant sind jedoch oft nur Cookies, die auch beim nächsten Browserstart noch zur Verfügung stehen. Man nennt diese Cookies auch permanent. Unendlich lang gültige Cookies gibt es nicht - und aufgrund der Mengenbeschränkung von 300 Stück ist das auch schwer zu realisieren. Aber Sie können Cookies ein Ablaufdatum mit auf den Weg geben, und diese werden dann so lange im Datenspeicher des Browsers gehalten.

Um das mit ASP zu realisieren, müssen Sie die Eigenschaft Expires des entsprechenden Cookies auf den gewünschten Datumswert setzen:

```

<%
 Response.Cookies("Name") = "Erik"
 Response.Cookies("Erik").Expires = "24.12.2001 23:45:56"
%>

```

Es stellt sich nun die Frage, wie lange dieser Cookie überhaupt gültig sein soll. Manche Leute empfehlen, das Ablaufdatum sehr weit in die Ferne zu setzen, beispielsweise ins Jahr 2037; das ist

jedoch relativ unfreundlich denjenigen Leuten gegenüber, die nur durch Zufall auf die Seite gestoßen sind und sicher nicht wieder so bald dorthin zurückkehren werden. Aus diesem Grund empfehlen wiederum andere Leute, das Ablaufdatum recht bald zu setzen, also beispielsweise zwei Monate nach Erstellung der Webseiten. Auch dieses Vorgehen bringt Nachteile mit sich, denn wenn Sie solch ein Datum in Ihren Code schreiben, müssen Sie diesen Code regelmäßig auf neue Datumswerte aktualisieren.

Das wohl beste Vorgehen vereint die beiden oben genannten Möglichkeiten. Das Ablaufdatum wird zwar recht bald gesetzt - beispielsweise zwei Monate in der Zukunft - aber es wird auf zwei Monate nach dem aktuellen Datum gesetzt. Wenn der Benutzer die Seiten erneut besucht, bekommt er einen neuen Cookie, der dann wiederum zwei Monate lang gültig ist und den alten Cookie überschreibt.

Die Umsetzung dieses Plans in ASP ist einfacher als Sie denken:

```
<%
 Response.Cookies("Name") = "Erik"
 Response.Cookies("Erik").Expires = Date + 60
%>
```

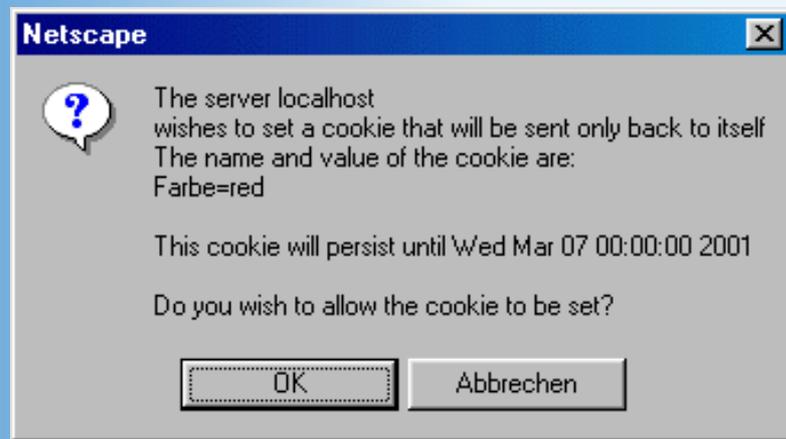
Der Ausdruck `Date + 60` liefert ein Datum, das 60 Tage nach dem aktuellen Datum liegt (wenn Sie Wert auf Sekundengenauigkeit legen, können Sie auch `Now + 60` verwenden). So haben Sie mit sehr wenig Aufwand ein stets akkurates Ablaufdatum. Mit diesem neuen Wissen lässt sich das Beispiel von zuvor mit der Lieblingsfarbe erweitern:

### Listing 18.18: `cookie_schreiben2.asp`

```
<%
 If Request.Form("submit") = "Abschicken" Then
 Response.Cookies("Farbe") = Request.Form("Farbe")
 Response.Cookies("Farbe").Expires = Date + 60
 End If
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Cookies</title>
</head>
<body bgcolor="white">
<h2>Cookies</h2>
<%
 If Request.Form("submit") = "Abschicken" Then
%>
Sie haben <% =Request.Form("Farbe") %>
als Ihre Lieblingsfarbe angegeben!
<%
 Else
%>
<form method="post">
<input type="text" name="Farbe" size="20" /> Lieblingsfarbe

<input type="submit" name="submit" value="Abschicken">
```

```
</form>
<%
 End If
%>
</body>
</html>
```



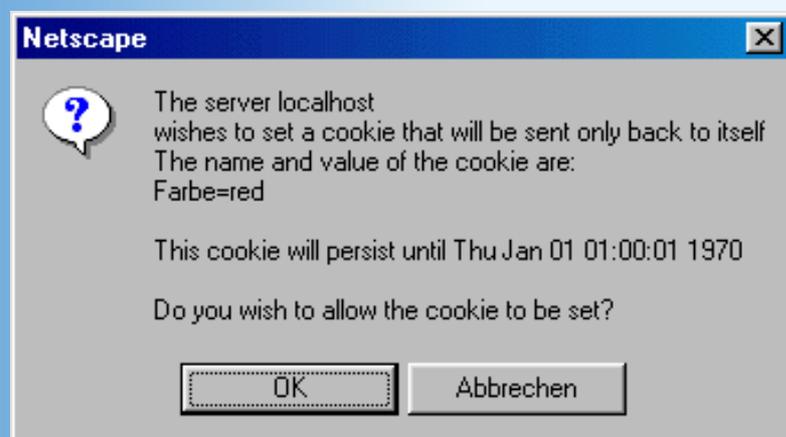
**Abbildung 18.21: Die Lieblingsfarbe wird drei Monate lang gespeichert**

## Cookies löschen

Um ein Cookie zu löschen, brauchen Sie auch keinen großen Aufwand zu treiben. Sie könnten natürlich den Wert des Cookies auf einen leeren String setzen, aber dann bleibt der Cookie (wenn auch ohne Wert) weiterhin im System des Surfers bestehen und belegt dort Speicher. Besser ist es, das Ablaufdatum des Cookies auf einen Wert in der Vergangenheit zu setzen; der Browser entfernt dann den Cookie automatisch:

```
<%
 Response.Cookies("Erik").Expires = Now - 1
%>
```

Im Browser sehen Sie dann - sofern Sie Warnmeldungen beim Eintreffen von Cookies aktiviert haben - etwas Merkwürdiges (siehe ): Obwohl Sie als Ablaufdatum den gestrigen Tag angegeben haben, schlägt der Browser als Ablaufdatum den 1. Januar 1970, 1 Uhr und 1 Sekunde vor.



**Abbildung 18.22: Das Ablaufdatum beim Löschen eines Cookies**

Das hat historische Gründe: Unter Unix/Linux wird die aktuelle Zeit als Anzahl der seit dem 1. Januar 1970 verstrichenen Millisekunden angegeben (auch *Epoche* genannt). Das vom Browser vorgeschlagene Ablaufdatum ist also eines der frühestmöglichen Datumswerte im System, auch wenn Sie ein anderes - in der Vergangenheit liegendes - Datum angegeben haben. Gelöscht wird das Cookie aber allemal.

## Cookies auslesen

Nachdem wir uns so lange über das Setzen von Cookies ausgelassen haben, werfen wir nun endlich einen Blick auf die andere Seite: Nämlich das Auslesen von Cookies. Wie bereits erwähnt: Wenn Sie serverseitig ein Cookie auslesen wollen, müssen Sie die HTTP- Anforderung des Clients untersuchen. Dazu ist unter ASP das Request-Objekt zuständig, welches Sie beispielsweise schon von den Formularen her kennen. Alle Cookies stehen in der Kollektion Request.Cookies, und Sie können beispielsweise so auf ein Cookie zugreifen:

```
<%
 Response.Write "Wert im Cookie 'Name': " & Request.Cookies("Name")
%>
```

Auch hier gibt es wieder eine kleine Abkürzung: Mit Request("Name") geht es auch! Zunächst wird untersucht, ob Request.Form("Name") existiert, dann ob es Request.QueryString("Name") gibt, und dann, ob Request.Cookies("Name") etwas liefert.



*Wenn Sie mit Formularen arbeiten, sollten Sie also die Abkürzung eher selten einsetzen, nicht dass Sie dann irgendwann mit einem gleichnamigen Cookie Probleme haben.*

## Praxisbeispiel

Zu Cookies gibt es mehrere Einsatzbeispiele. Beispielsweise kann in einem Cookie der aktuelle Warenkorb bei einer Shoppinglösung gespeichert werden, oder in einem geschützten Bereich der Login-Name des Benutzers. Ebenso kann gezählt werden, wie oft der Benutzer schon auf der aktuellen Seite war, und ob er ein bestimmtes JavaScript- Popup-Fenster schon gesehen hat. Wir wollen hier aber eine andere Anwendungsmöglichkeit zeigen, und zwar, wie Sie mit Cookies recht schnell eine personalisierte Website erstellen können. Im einfachsten Beispiel kommen wir auf den vorherigen Code mit der Lieblingsfarbe zurück. Der dort eingegebene Wert wird als Hintergrundfarbe auf jeder Seite verwendet. Wie das Cookie gespeichert wird, haben Sie bereits vorher gesehen; jetzt werfen wir einen Blick auf die Ausgabeseite. Der Wert im Cookie wird einfach als bgcolor-Attribut des <body>-Tags eingesetzt.



*Ein paar Sicherheitsvorkehrungen haben wir eingebaut. Ist das Cookie nicht gesetzt*

(oder leer), wird als Hintergrundfarbe Weiß eingesetzt. Ebenso werden öffnende und schließende spitze Klammern herausgefiltert, denn die haben ja in HTML eine besondere Bedeutung (und so könnte ein böswilliger Surfer über den Umweg mit der Hintergrundfarbe beispielsweise JavaScript-Code in die Seiten einbauen).

### Listing 18.19: Lieblingsfarbe.asp

```
<%
Option Explicit
Dim strFarbe
strFarbe = Request.Cookies("Farbe")
strFarbe = Replace(strFarbe, "<", "")
strFarbe = Replace(strFarbe, ">", "")
If strFarbe = "" Then
 strFarbe = "white"
End If
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Personalisierung</title>
</head>
<body bgcolor="<% =strFarbe %>">
<h2>Willkommen auf Ihrer personalisierten Seite!</h2>
<p>Ihre Lieblingsfarbe ist: <% =strFarbe %>.</p>
</body>
</html>
```



Abbildung 18.23: Die Lieblingsfarbe wird als Hintergrundfarbe verwendet

Ihren eigenen Ideen sind somit kaum mehr Grenzen gesetzt. Wenn Sie sich weiter in ASP

earbeiten wollen, am Anfang dieses Tages finden Sie einen Literaturhinweis. Auf alle Fälle sollten Sie sich noch die VBScript-Dokumentation von Microsoft herunterladen, denn diese enthält eine komplette Befehlsübersicht und ist somit eine unverzichtbare Hilfe für den ASP-Programmierer. Sie finden die Dokumentation momentan unter der URL

<http://msdn.microsoft.com/scripting/vbscript/download/vbsdoc.exe>.

Und noch ein Hinweis: Sollte Ihnen dieser Tag recht lange vorgekommen sein - keine Bange, morgen geht es etwas ruhiger zu, auch wenn wir ein komplett neues Thema betrachten: PHP, die Open Source-Konkurrenz zu ASP!

## 18.7 Fragen und Antworten

**Frage:**

**Wie bekomme ich in ASP heraus, von welcher Seite der Benutzer herkam? Welche IP- Adresse er hat?**

*Antwort:*

*All dies (und noch viel mehr) steht in der Kollektion Request.ServerVariables. Sie können diese mit For-Each durchschreiten und sich die interessanten Werte ausgeben lassen. Um die Frage zu beantworten: Verwenden Sie Request.ServerVariables("HTTP\_REFERER") bzw. Request.ServerVariables("REMOTE\_ADDR").*

**Frage:**

**Wie kann ich einen numerischen Wert in einen Integerwert umwandeln und umgekehrt?**

*Antwort:*

*ASP bietet dazu Funktionen an, die Sie in der Dokumentation finden (unter »C«). Besonders einfach - wenn auch etwas unsauber - geht es folgendermaßen: Integers werden zu Strings, wenn Sie die Integervariable mit »« konkatenieren. Strings werden zu Integer-Werten (oder auch zu Floats), wenn Sie sie mit 1 multiplizieren.*

**Frage:**

**Kann ich mit ASP ein wenig HTML-Code ausgeben, und dann nach ein paar Sekunden den Benutzer auf eine andere Seite weiterleiten?**

*Antwort:*

*Nein, das geht nur mit clientseitigen Mitteln. In den vorhergehenden beiden Wochen haben Sie dazu Mittel und Wege kennen gelernt, sowohl mit HTML als auch mit JavaScript.*

**Frage:**

**Kann ich mit ASP die Bildschirmauflösung des Benutzers herausbekommen?**

*Antwort:*

*Nein! ASP ist eine serverseitige Technologie, und damit können Sie solche clientseitigen Informationen nicht erhalten. Verwenden Sie auch hier JavaScript.*

**Frage:**

**Ich will eine Datei im aktuellen Verzeichnis erstellen. Wie gehe ich vor, wenn ich nicht den absoluten Pfad auf der Festplatte eingeben will (denn dann kann ich recht schlecht mit der Anwendung auf einen anderen Server umziehen)?**

*Antwort:*  
Durch `Server.MapPath("datei.txt")` bekommen Sie den absoluten Pfad der Datei `datei.txt` im aktuellen (virtuellen) Verzeichnis heraus.

**Frage:**  
**Immer, wenn ich eine Datei erstellen will, erhalte ich als Meldung »Permission denied« oder »Zugriff verweigert«. Was kann ich tun?**

*Antwort:*  
Sie haben offensichtlich keine Schreibrechte in das aktuelle Verzeichnis oder die aktuelle Datei. Bei Dateien überprüfen Sie bitte, ob der Schreibschutz aktiviert ist; andernfalls bitten Sie Ihren Administrator (oder Provider) um Hilfe. Viele Provider ermöglichen nur auf Nachfrage Schreibrechte in bestimmte Verzeichnisse.

## 18.8 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

1. Ist ASP eine Programmiersprache?
2. Wozu dient Option Explicit?
3. Ist das Gleichheitszeichen im VBScript ein Vergleichs- oder ein Zuweisungsoperator?
4. Was ist der Hauptunterschied zwischen If-Then und Select Case?
5. Mit welchen Schleifen können Sie besonders bequem auf alle Arrayelemente zugreifen?
6. Was passiert, wenn Sie das action-Attribut eines Formulars leer lassen?
7. Wie greifen Sie auf alle ausgewählten Elemente einer Auswahlliste zu, die mit `<select multiple>` erstellt worden ist?
8. Wozu ist der zweite Parameter von `OpenTextFile` gut?
9. Was ist der Hauptunterschied beim Lesen und beim Setzen von Cookies?

### Übungen

1. Begrüßen Sie auf einer ASP-Seite Ihre Besucher je nach Tageszeit (»Guten Morgen« etc.)!
2. Erweitern Sie die Vollständigkeitsüberprüfung des Formulars aus diesem Kapitel! Momentan ist es so, dass bei nicht vollständiger Ausfüllung das Formular erneut angezeigt wird, aber komplett leer. Sorgen Sie dafür, dass bereits eingegebene Werte stehen bleiben.
3. Fassen Sie die Listings zum Thema Personalisierung zusammen: Auf einer ASP-Seite erscheint entweder die angegebene Hintergrundfarbe, oder der Benutzer wird zu ihrer Eingabe aufgefordert.

Die Aufteilung der Kontinente ist nicht einheitlich. Manche Leute trennen den Kontinent Amerika in Nord- und Südamerika auf, manche wiederum fassen Australien und die Antarktis zum Kontinent Ozeanien zusammen. Bayern ist aber in keiner Quelle als eigener Kontinent zu finden.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

## Tag 19

# PHP - Einführung

Während Sie gestern etwas völlig Neues kennen gelernt haben, nämlich die Microsoft'sche Definition von serverseitiger Programmierung, kommen wir heute zum Hauptkonkurrenten, nämlich PHP. Es handelt sich hierbei um eine weitere serverseitige Programmiersprache, die insbesondere in letzter Zeit Furore macht. Genau so wie gestern erhalten Sie zunächst einen Crash-Kurs in die Sprachsyntax, und dann sehen Sie, wie einige Standardaufgaben mit PHP gelöst werden können. Auch hier gilt leider wieder: Für einen kompletten Kurs fehlt uns leider die Zeit bzw. der Platz. Bei PHP gilt das noch mehr als bei ASP, denn die Sprache kennt weit über 1000 verschiedene Funktionen! Aus diesem Grunde auch hier Literaturhinweise: Für Einsteiger empfiehlt sich das (allerdings recht Linux-lastige) Buch »Jetzt lerne ich PHP 4«, oder das Standardwerk, und nebenbei auch das erste deutschsprachige Buch überhaupt zu dem Thema (von Mitgliedern der PHP Documentation Group mit verfasst und enthält damit auch die offizielle Funktionskurzübersicht), »PHP 4«, ISBN 3-8272-5877-4.

Im Anhang finden Sie eine Installationsanleitung, und zwar sowohl unter Linux, als auch unter Windows (und hier auch mit verschiedenen Webservern). Weiterführende Hinweise finden Sie auch im World Wide Web, auf der offiziellen PHP-Seite unter [www.php.net](http://www.php.net). Das Online-Manual, immer auf dem aktuellen Stand, mit einer Übersicht über alle Funktionen sowie mit Beispielen und Hinweisen, gibt es unter [www.php.net/manual](http://www.php.net/manual).

Die Themen heute:

- Was ist PHP?
- Variablen
- Operatoren
- Schleifen und Fallunterscheidungen
- Mit Datumswerten arbeiten
- Formularhandling
- File-Uploads
- Dateizugriffe
- Cookies lesen und setzen

## 19.1 Was ist eigentlich PHP?

Fast jede Programmiersprache hat eine Legende, und bei PHP ist das nicht anders. Ein gewisser Rasmus Lerdorf hat eine Reihe von Tools geschrieben, die ihn in seiner täglichen Arbeit unterstützen sollten. Nun ja, er hat diese Tools der Öffentlichkeit zur Verfügung gestellt, und

Mundpropaganda und geschicktes Marketing taten ihr weiteres. Ursprünglich hieß das ganze PHP/FI, dann wurde es in PHP umbenannt, und Version 3 der Sprache, oder auch PHP 3, wurde veröffentlicht. Im Jahr 2000 erschien dann PHP 4, eine komplett neu geschriebene PHP-Version, die intern einen völlig neuen Aufbau aufwies, aber dennoch (größtenteils) abwärtskompatibel zu Version 3 war. Die im Hintergrund operierende PHP-Engine heißt *Zend*, nach den Namen der beiden Hauptentwickler.

Ein weltweites Netz von Entwicklern entwickelte damals (und auch heute noch) an der Sprache, und zwar nach dem Prinzip *Open Source*: Der Quellcode steht zur Verfügung, und jeder kann die Sprache weiterentwickeln, neuen Code hinzufügen, der dann von den Entwicklern begutachtet und eventuell eingebaut wird. Das Prinzip ist also dasselbe wie beim Betriebssystem Linux.

Schon hier wird deutlich, dass PHP eine klare Konkurrenzposition zu ASP eingenommen hat. Genauso, wie der Quellcode der Microsoft-Betriebssysteme (noch) nicht offengelegt worden ist, steht auch der Quellcode des ASP-Interpreters der Öffentlichkeit nicht zur Verfügung. Wir wollen uns damit keineswegs aufs Glatteis begeben und behaupten, die eine oder die andere Variante sei die bessere. Dieser Hintergrund ist aber wichtig, um einige Hintergründe der Sprache besser zu verstehen.

PHP wurde unter Unix/Linux entwickelt, und zielte auch lange Zeit auf diese Plattformen ab. Irgendwann gab es eine Windows-Portierung, aber auch die wurde eine Zeitlang von den Kernentwicklern nur müde belächelt oder als (nicht ganz notwendiges) Beiprodukt gesehen. Windows-Nutzer wurden in den PHP-Diskussionslisten anfangs genauso behandelt. Sie sehen also: Ideologische Glaubenskriege sind auch hier nie spurlos vorübergezogen.

Aber auch andere Besonderheiten hielten den Aufstieg von PHP in die erste Liga der kommerziell eingesetzten Programmiersprachen auf. Die Dokumentation war lange Zeit ein großer Schwachpunkt, denn Entwickler dokumentieren ja prinzipiell ungern, und ein gewisser Mangel an Professionalität war beim PHP-Projekt lange Zeit nicht wegzudiskutieren. Entwickler waren auf Kritik oder Änderungswünsche nicht besonders gut zu sprechen, und der Ton in den Diskussionslisten wurde teils sehr scharf. Technische Entscheider, die sich überlegen mussten, ob sie PHP einsetzen wollten, fanden also genügend Argumente, die Technologie zunächst ad acta zu legen.

So nach und nach hat sich die Situation aber verbessert. Heutzutage stehen Windows-Binaries einer neuen PHP-Version nur Stunden nach Veröffentlichung des Quellcodes einer neuen Version zur Verfügung. Die PHP Documentation Group (unter Leitung von Egon Schmid) sorgt dafür, dass die Dokumentation immer auf dem neuesten Stand ist. Mittlerweile wurde erkannt, dass PHP nur dann den Durchbruch schaffen kann, wenn man ideologische Mauern einreißt und versucht, eine möglichst große Zielgruppe zu erreichen. Denn oftmals ist es so: Entwickler greifen gerne zu ihren Windows-Maschinen für die Programmierung, auch wenn der endgültige Zielsystem dann unter Linux läuft. Inzwischen gibt es sogar eine eigene Website, die Windows-Binaries auch von Entwickler-Versionen (Vorabversionen) von PHP anbietet: [www.php4win.de](http://www.php4win.de).

Die ursprüngliche Standardplattform von PHP ist LAMP: Linux als Betriebssystem, Apache als Webserver, MySQL als Datenbank, und natürlich PHP. Anfangs wurde oft empfohlen, unter Windows analog von WAMP zu sprechen: Windows, Apache, MySQL, PHP. Das ist in den meisten Fällen jedoch grober Unfug. Viele Leute, die sich mit der Installation von Apache unter

Windows abmühen, haben gar keine so großen Anforderungen an ihren Webserver, so dass der Personal Web Server von Microsoft völlig genügt. Der Internet Information Server von Microsoft ist dem Apache sogar weitestgehend ebenbürtig. Zu beiden Servern finden Sie im Anhang übrigens eine Installationsanleitung.

Nun ein Wort zur verwendeten Datenbank: MySQL ist gratis, und unter Linux weit verbreitet. Unter Windows gibt es jedoch einige Alternativen. Wer keine großen Anforderungen hat, kann Access verwenden; für die Beispiele in diesem Buch (morgen gehen wir auf die Datenbankprogrammierung ein) wird dazu nicht mal Access selbst benötigt! Professionellere Anwendungen verwenden als Datenbank Oracle oder Microsoft SQL Server, zwei Produkte, die insbesondere bei größeren Datenmengen MySQL überlegen sind. Man kann diese Datenbanken inzwischen sogar über eigene PHP- Funktionen ansteuern, aber unter Windows gibt es für den Datenbankzugriff einen kleinsten gemeinsamen Nenner: ODBC, *Open Database Connectivity*. Hiermit kann quasi jede Datenquelle unter Windows mit einem einheitlichen Befehlssatz angesteuert werden.

Warum dennoch so viele Leute auf einmal wie wild angefangen haben, auf Ihren Windows-Maschinen Apache und MySQL zu installieren, und dann laut geflucht haben, weil es doch etwas komplizierter war als die »Klicki-Bunti«-Welt von Windows? Nun, viele Bücher waren sehr Linux-lastig, und Windows und/oder ODBC waren Fremdwörter. Hier eine gute Nachricht: Wir gehen sowohl auf Linux als auch auf Windows ein, Sie können also die für Ihre Anforderungen optimale Lösung wählen und sind nicht durch die Vorgaben Ihrer Autoren eingegrenzt.

Legen wir also los, legen Sie die Sitzgurte an, und halten Sie sich fest. Im Folgenden erhalten Sie eine Spracheinführung im Schnelldurchlauf und dann eine Reihe von Anwendungsbeispielen. Und noch ein Wort zum Umfang des heutigen Tages (in Sachen Buchseiten) im Vergleich zu gestern. Ja, es sind weniger Seiten, aber das heißt weder, dass wir ASP über PHP bevorzugen oder umgekehrt, noch dass die eine oder die andere Sprache schwieriger oder leichter zu erlernen sei. Der Grund ist folgender: PHP orientiert sich von der Syntax an JavaScript und Perl, und insbesondere mit ersterem sollten Sie inzwischen sehr vertraut sein. Wir können also viele einleitende Dinge kürzer fassen. Der Umfang der Anwendungsbeispiele ist ungefähr gleich zum ASP-Teil, Sie verpassen also nichts. Um genau zu sein, werden Sie einige der Beispiele sogar leicht modifiziert wiederfinden; dies ermöglicht es Ihnen, einen direkten Vergleich zwischen den beiden Sprachen zu ziehen.

Als allerletzte Vorbereitung sollten Sie noch einmal sicherstellen, dass PHP auf Ihrem Webserver korrekt eingerichtet ist. Erstellen Sie eine Datei mit folgendem Inhalt:

### **Listing 19.1: phpinfo.php**

```
<?php
 phpinfo();
?>
```

Rufen Sie diese Datei über ihre Server-URL auf, also beispielsweise als *http://localhost/skriptname.php*, und vergleichen Sie die Anzeige im Webbrowser mit der aus Abbildung 19.1. Sofern Sie eine gewisse Ähnlichkeit erkennen können, haben Sie (zunächst) alles richtig gemacht und können loslegen. Falls nicht, werfen Sie noch einmal einen Blick auf die

Installationsanleitung im Anhang, oder suchen Sie im Online-Manual nach Hilfe.

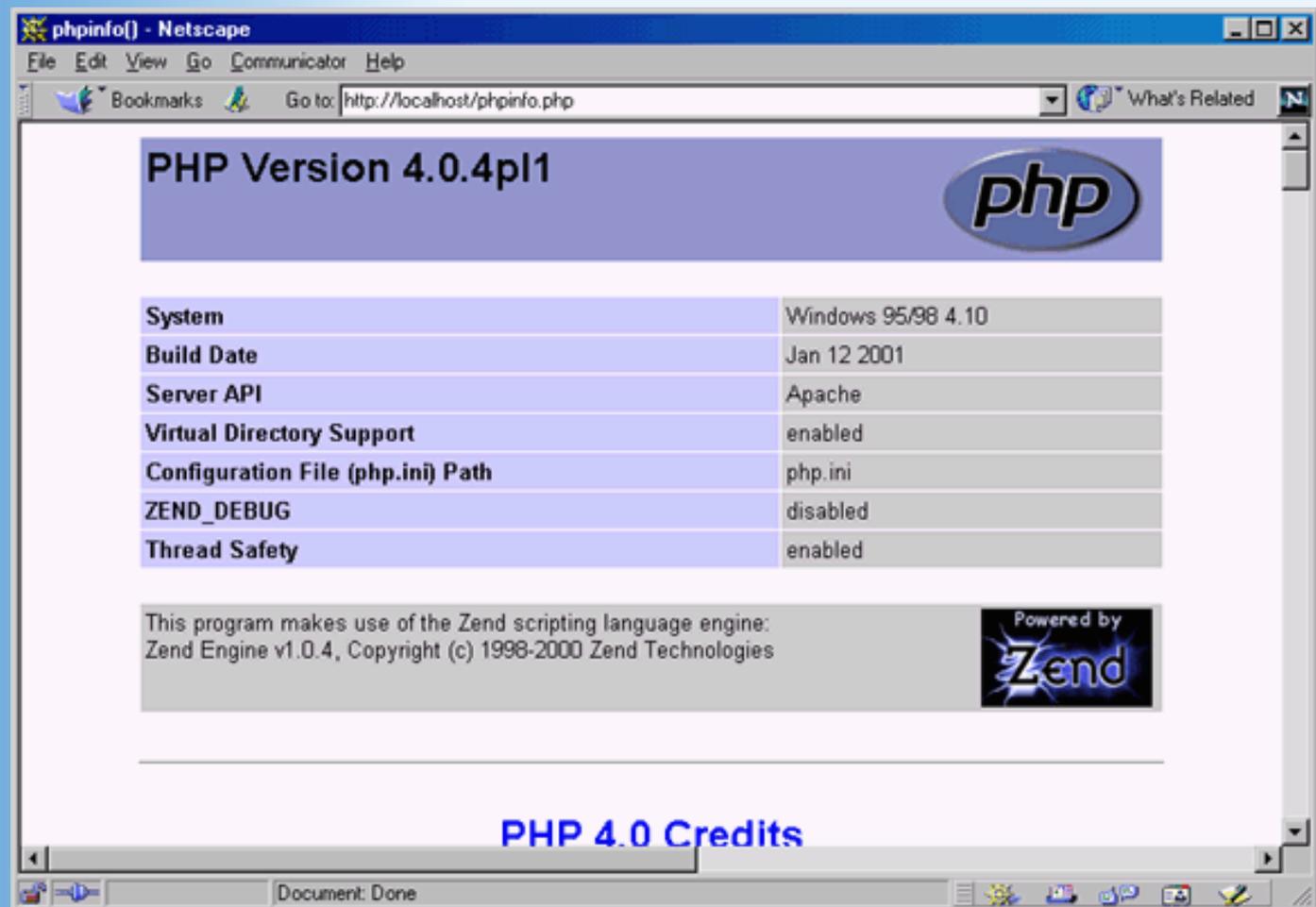


Abbildung 19.1: PHP ist korrekt installiert und eingerichtet worden

## 19.2 Spracheinführung

Steigen wir also ohne weitere Umschweife in die PHP-Programmierung ein. Vieles, was Sie hier sehen werden, wird Ihnen nur allzu bekannt vorkommen. PHP ähnelt (nicht ohne Absicht) Perl sehr stark, und auch JavaScript-Kenner werden das eine oder andere Déjà-vu- Erlebnis haben. Also, frisch ans Werk!

### Einbau

Was bei ASP noch `<%` und `%>` war, ist bei PHP `<?php` und `?>`. Sie sehen: Auch hier wird der Programmcode direkt in ein HTML-Dokument eingebettet, was unter anderem den Vorteil hat, dass Sie PHP-Skripte in praktisch jedes beliebige Verzeichnis legen können, im Gegensatz zu Perl, wo Sie zumeist alles in *cgi-bin* legen müssen, was die Verlinkung auf andere Seiten etwas erschwert.



*Wohl auch aus diesem Grund rechnen manche Experten damit, dass PHP mittelfristig Perl als Web-Skriptsprache ablösen wird.*

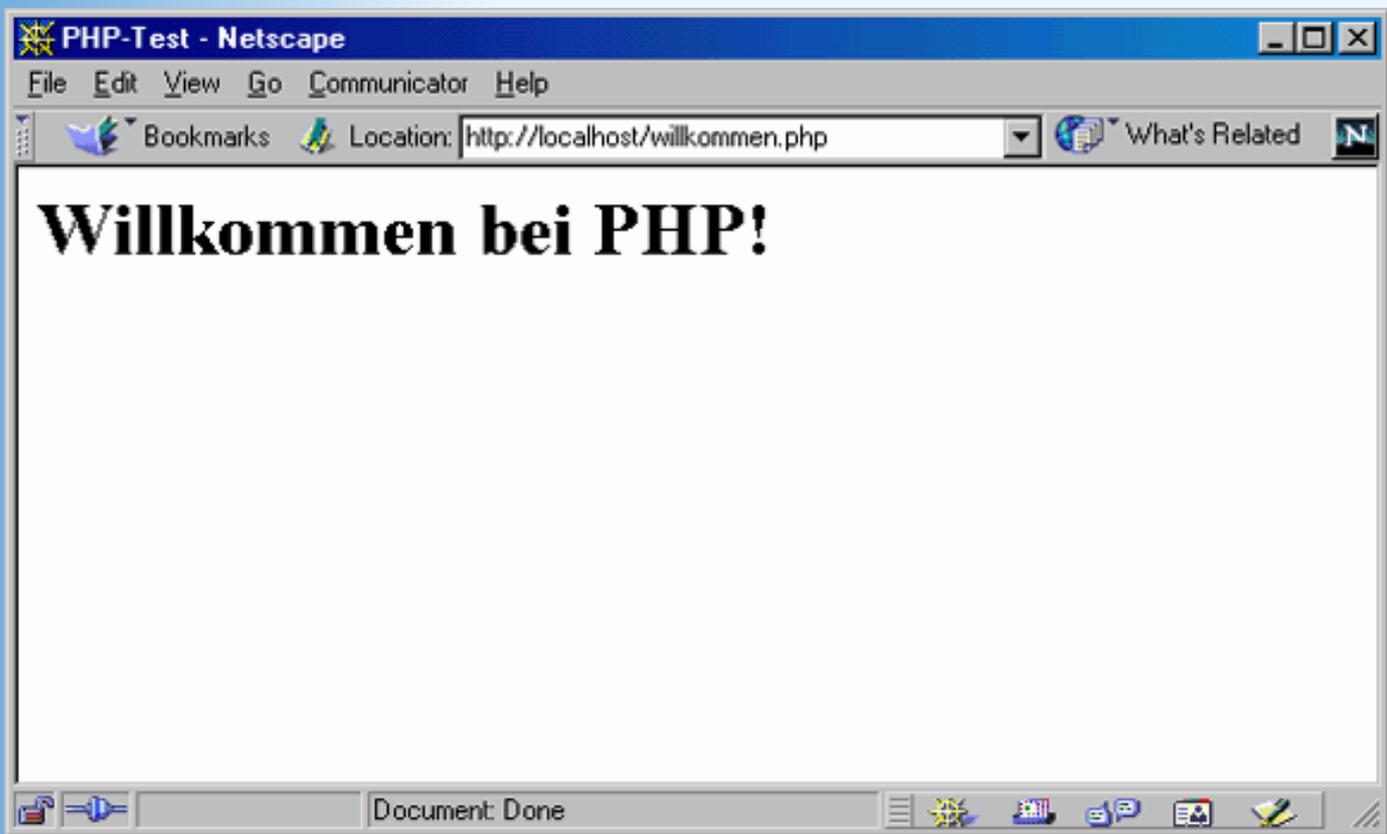
Es gibt auch andere Möglichkeiten, PHP-Code vom HTML-Code abzugrenzen. Eine Möglichkeit besteht in der Verwendung von `<script language="php"> ... </script>`, aber - durch entsprechende Modifikationen in der `php.ini` - ist auch `<? ... ?>` und sogar `<% ... %>` möglich. Standard ist und bleibt aber `<?php ... ?>`, woran wir uns heute auch halten wollen und werden. Sehr häufig findet man außerdem noch oben erwähntes `<? ... ?>`.

Hier ein erstes, einfaches, PHP-Skript:

### **Listing 19.2: willkommen.php**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>PHP-Test</title>
</head>
<body bgcolor="white">
<?php
 print "<h1>";
 print "Willkommen bei PHP!";
 print "</h1>";
?>
</body>
</html>
```

Im Browser sollten dieses PHP-Skript nun zu einer ähnlichen Ausgabe führen.



**Abbildung 19.2: Das erste Skript im Webbrowser**

Ihnen wird einiges aufgefallen sein:

- Mit print können Texte ausgegeben werden, die dann an den Webbrowser geschickt werden.
- Jede Anweisung wird durch ein Semikolon beendet. JavaScript- und Perl-Kenner vermuten hier gleich richtig: Man kann auch mehrere Anweisungen in eine Zeile schreiben.

Ein Blick auf den Quellcode der resultierenden Seite bestätigt zumindest die erste Vermutung:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>PHP-Test</title>
</head>
<body bgcolor="white">
<h1>Willkommen bei PHP!</h1></body>
</html>
```



*Zur Erinnerung: Sie können den Quelltext einsehen, indem Sie beim Netscape Navigator **View/Page Source** bzw. **Anzeige/Seitenquelltext** wählen, beim Internet*

Explorer lautet der Menübefehl **Ansicht/Quelltext** anzeigen.

Und noch ein Wort zur Klammerung: Es ist auch möglich, den Parameter für print in Klammern zu schreiben, also print "irgendein Text";. Bei nur einem Parameter unterlässt man das in der Praxis aber, wohl auch deswegen, weil die print-Anweisung naturgemäß sehr häufig verwendet wird.

Kommentare gibt es in PHP natürlich auch. Zwei davon kennen Sie von JavaScript, einen davon von Perl. Alles weitere finden Sie in !

Kommentarzeichen	Beispiel	Beschreibung
//	// Kommentar	einzeiliger Kommentar
/* ... */	/* Kommentar Kommentar */	mehrzeiliger Kommentar
#	# Kommentar	einzeiliger Kommentar

**Tabelle 19.1: Die Kommentare in PHP**

## Variablen

Am Anfang war das Wort... in unserem Falle sind es aber die Variablen, Grundbaustein für die Programmierung. Eine Variable wird - genauso wie in Perl - durch ein vorangestelltes Dollarzeichen gekennzeichnet. Für die Namensgebung der Variablen gibt es die folgenden Regeln und Tipps:

- Verwenden Sie am Besten nur Buchstaben, sowie den Unterstrich (\_)
- Wenn Sie auch Ziffern verwenden, nicht am Anfang des Variablennamens!
- Überlegen Sie sich kurze, prägnante, aber eindeutige Bezeichner für Ihre Variablen!
- PHP unterscheidet zwischen Groß- und Kleinschreibung! Die Variable \$name ist also nicht identisch zu \$NAME!

## Zuweisungen

Als Zuweisungsoperator fungiert auch hier das Gleichheitszeichen. Links davon steht der Variablenname, rechts davon der Wert (oft auch in Form eines Ausdrucks), den die Variable annehmen soll.

```
<?php
 $ping = "pong";
 $pingpong = $ping;
?>
```

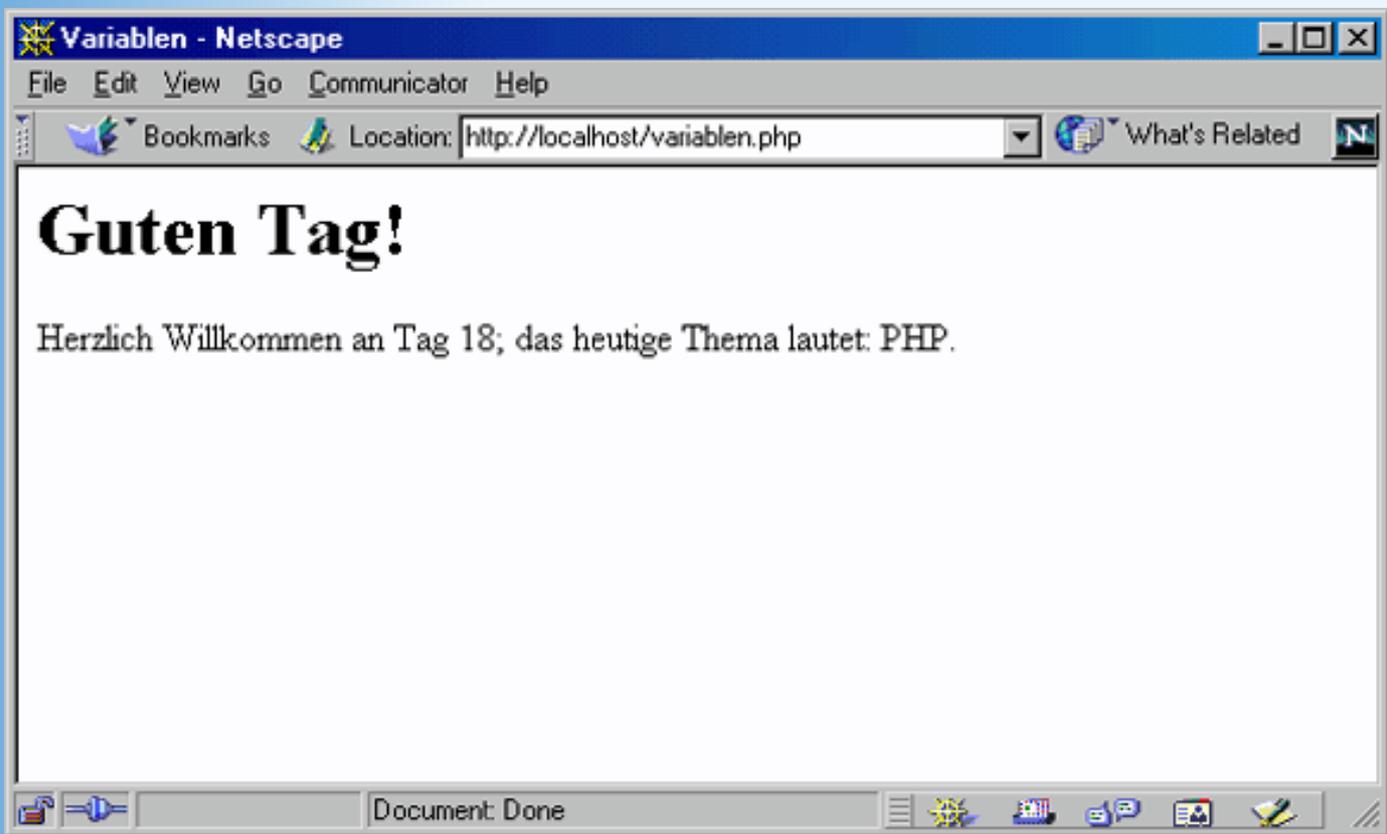
Nach Ausführung dieses Codes haben sowohl die Variable \$ping als auch die Variable \$pingpong denselben Wert: »pong«.

## Ausgabe

Die Ausgabe einer Variablen funktioniert mit der Anweisung `print`, die Sie bereits zuvor im Abschnitt 19.2.1 kennen gelernt haben. Als Parameter wird einfach die Variable übergeben:

### Listing 19.3: `variablen.php`

```
<?php
 $begrueessung = "Guten Tag!";
 $tag = 18;
 $thema = "PHP";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Variablen</title>
</head>
<body background="white">
<h1>
<?php
 print $begrueessung;
?>
</h1>
Herzlich Willkommen an Tag
<?php
 print $tag;
?>;
das heutige Thema lautet:
<?php
 print $thema;
?>.
</body>
</html>
```



**Abbildung 19.3: Der Inhalt der Variablen wird ausgegeben**



*Es gibt noch eine Alternative zur print-Anweisung, auf die Sie im Web wohl hin und wieder stoßen werden: echo. Die Unterschiede zwischen den beiden Anweisungen sind aber so gering, dass Sie Ihnen in der Regel gar nicht auffallen werden (eine der beiden Funktionen hat einen - überflüssigen - Rückgabewert).*

Genauso wie bei ASP gibt es bei PHP auch eine Kurzform für die print-Anweisung, und zwar das Gleichheitszeichen in Verbindung mit <? (also nicht <?php!). Sie finden diese Abkürzung aber in der Praxis sehr, sehr selten im Einsatz. Aus diesem Grund sollten Sie sie auch nur sehr, sehr selten einsetzen. Rechnen Sie immer damit, dass andere Leute (beispielsweise Kollegen) Ihren Code einmal überarbeiten müssen, und da sollten Sie sich an allgemein akzeptierte, wenn auch ungeschriebene Standards halten.

Hier noch einmal das Listing von oben, unter Verwendung der Kurzform:

#### **Listing 19.4: variablen\_kurz.php**

```
<?php
 $begrueßung = "Guten Tag!";
 $tag = 19;
 $thema = "PHP";
?>
```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Variablen</title>
</head>
<body background="white">
<h1><?=$begruessung; ?></h1>
Herzlich Willkommen an Tag <?=$tag; ?>;
das heutige Thema lautet: <?=$thema; ?>.
</body>
</html>

```

## Strings und Interpolation

Strings werden (unter anderem) durch doppelte Anführungszeichen umfasst, also analog zu allen Sprachen, die Sie bisher kennen gelernt haben. Alternativ dazu (und das ist ein echter Unterschied zu ASP) können Sie einen String auch durch einfache Anführungszeichen umgeben:

```

<?php
 $d = "Dynamic";
 $wp = 'Web-Publishing';
?>

```

Innerhalb von doppelten Anführungszeichen gibt es aber eine Besonderheit: Variablenamen werden durch ihren aktuellen Wert ersetzt! Werfen Sie einen Blick auf das folgende Beispiel:

### Listing 19.5: interpolation.php

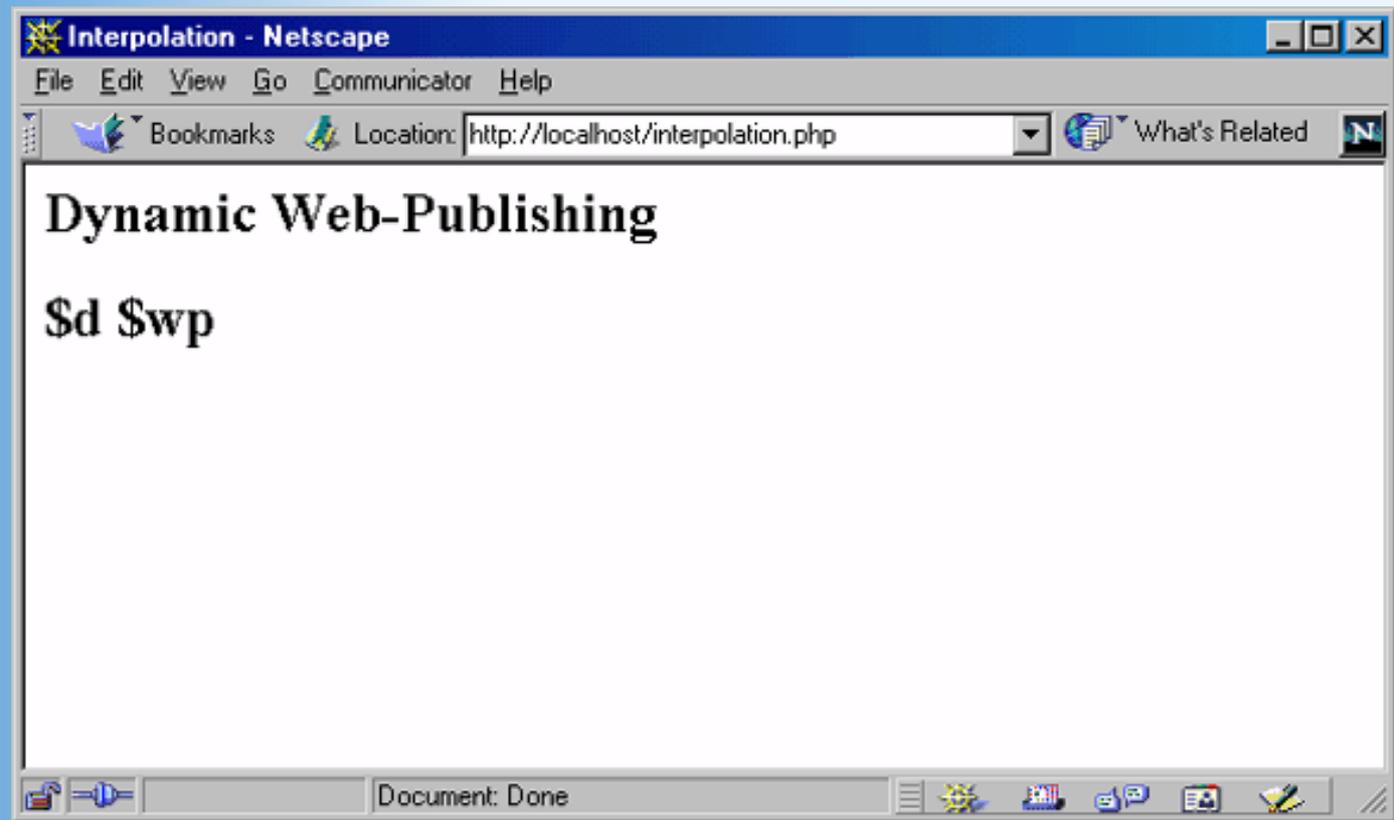
```

<?php
 $d = "Dynamic";
 $wp = 'Web-Publishing';
 $dwp1 = "$d $wp";
 $dwp2 = '$d $wp';
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Interpolation</title>
</head>
<body bgcolor="white">
<h2><?php print $dwp1; ?></h2>
<h2><?php print $dwp2; ?></h2>
</body>
</html>

```

Laden Sie dieses Listing in Ihren Webbrowser, und Sie werden den Unterschied zwischen

einfachen und doppelten Anführungszeichen sehen (siehe auch ). Bei doppelten Anführungszeichen (erste Ausgabezeile) werden Variablenwerte *interpoliert*, bei einfachen Anführungszeichen (zweite Ausgabezeile) nicht.



**Abbildung 19.4: Einmal Interpolation (oben), einmal nicht (unten)**

Naheliegende Frage - wie ist es dann möglich, das Dollarzeichen innerhalb eines Strings auszugeben? Nun, verwenden Sie einfache Anführungszeichen (wie im vorherigen Listing zu sehen), oder stellen Sie dem Dollarzeichen einen Backslash voran. Dadurch »entwerten« Sie das Sonderzeichen. Auch dies ist Ihnen bereits aus den Tagen zu JavaScript und Perl bekannt.

```
<?php
 $sichtbar = "L'êtat, c'est moi";
 $unsichtbar = "\$sichtbar";
?>
```

Die Variable `$unsichtbar` enthält jetzt die Zeichenkette »`$sichtbar`«, und nicht den Wert der Variablen `$sichtbar`.

Einen Backslash können Sie innerhalb von doppelten Anführungszeichen natürlich auch entwerten - durch einen vorangestellten Backslash:

```
<?php
 print "Die php.ini liegt unter c:\\windows";
?>
```

Dieses Codestück gibt »Die php.ini liegt unter c:\windows« aus.

Eine weitere besondere Form der Entwertung findet statt, wenn Sie den Backslash vor ein doppeltes Anführungszeichen setzen. Innerhalb eines Strings wird dadurch ein doppeltes Anführungszeichen als Text interpretiert und nicht als Ende des Strings:

```
<?php
 print "Wofür steht \"PHP\" eigentlich?";
?>
```

Durch den Backslash können nicht nur Sonderzeichen entwertet werden, sondern einige Zeichen erhalten noch eine besondere Bedeutung:

- Mit `\n` können Sie einen Zeilensprung veranlassen (*new line*)
- Durch `\r` geben Sie einen Wagenrücklauf (*carriage return*)
- Der Code `\t` entspricht einem Tabulator (Tab)

## Weitere Variablentypen

Neben Strings gibt es natürlich auch die numerischen Variablen, also entweder ganzzahlige oder Fließkommawerte. Beachten Sie hier, dass es - wie bei fast allen Programmiersprachen - kein (deutsches) Dezimalkomma gibt, sondern einen (insbes. amerikanischen) Dezimalpunkt!

```
<?php
 $php3 = 1997;
 $euler = 2.718281828;
?>
```

Des Weiteren sind noch die booleschen Variablen erwähnenswert. Sie kennen zwei Zustände, `true` (wahr) und `false` (falsch).

```
<?php
 $t = true;
 $f = false;
?>
```

PHP ist keine typisierte Sprache, das heißt der Typ einer Variablen ist nicht fix, er kann sich also zur Laufzeit des Skripts ohne Weiteres ändern. Ein einfaches Beispiel illustriert dies:

```
<?php
 $euler = 2.718281828; //Variable ist ein Fließkommawert
 $euler = "etwa zwei Komma sieben"; //Variable ist jetzt ein String
?>
```

## Arrays

PHP bietet zwei Arten von Arrays. Die erste Art orientiert sich an der klassischen Vorstellung von Arrays: Über einen numerischen Index wird auf ein Array-Element zugegriffen. Der niedrigste Index hat auch hier den Wert 0, und der Index selbst steht in eckigen Klammern:

```
<?php
 $kontinente[0] = "Amerika";
 $kontinente[1] = "Afrika";
 $kontinente[2] = "Asien";
 $kontinente[3] = "Europa";
 $kontinente[4] = "Antarktis";
 $kontinente[5] = "Australien";
?>
```

Sie können sich ein wenig Schreibarbeit sparen, indem Sie die Funktion `array` verwenden. Als Parameter übergeben Sie die einzelnen Werte, die im Array gespeichert werden sollen. Der erste Parameter bekommt dann den Index 0, der zweite den Index 1, und so weiter:

```
<?php
 $kontinente = array("Amerika", "Afrika", "Asien", "Europa",
 "Antarktis", "Australien");
?>
```



*Sie können - wie zu sehen - längere Codezeilen beliebig umbrechen. Ausnahme: Innerhalb von Funktionsnamen oder von Werten wie etwa Strings.*

Die zweite Art von Array sind *assoziative* Arrays, also Arrays, bei denen als Index (oder auch *Schlüssel*) eine beliebige Zeichenkette fungiert und nicht notwendigerweise ein numerischer Wert. Achten Sie bei der Zuweisung aber darauf, dass die Indizes eindeutig sind, denn sonst überschreiben Sie den alten Wert!

```
<?php
 $hauptstadt["Deutschland"] = "Berlin";
 $hauptstadt["Ägypten"] = "Kairo";
 $hauptstadt["China"] = "Peking";
?>
```

Auch hier gibt es eine Möglichkeit, sich ein wenig Tipparbeit zu ersparen. Verwenden Sie erneut die `array`-Funktion, aber geben Sie die Schlüssel mit an, und trennen Sie sie mit `=>` von den entsprechenden Werten.

```
<?php
 $hauptstadt = array("Deutschland"=>"Berlin", "Ägypten"=>"Kairo",
 "China"=>"Peking");
?>
```

Im Zusammenhang mit Arrays ist noch die Funktion `count` interessant, welche die Anzahl der Elemente im Array zurückliefert. In Tabelle 19.1 sehen Sie die Rückgabewerte von `count` für die

exemplarischen Arrays von oben.

Ausdruck	Wert
count(\$kontinente)	6
count(\$hauptstadt)	3

**Tabelle 19.2: Die Rückgabewerte von count für die exemplarischen Arrays**

## Operationen und Operatoren

Variablen einfach statische Werte zuzuweisen ist langweilig. Die Programmierung lebt davon, dass man mit Variablen rechnen kann. Je nach Variablentyp gibt es unterschiedliche Operatoren, die Sie einsetzen können.

### Arithmetische Operatoren

Auch heute werden Sie wohl wieder einige Déja-vu-Erlebnisse haben. Die meisten Operatoren kennen Sie noch aus der Grundschule (auch wenn die dort etwas anders aussahen, beispielsweise Malpunkt statt Stern), und den Rest haben Sie bei irgendeiner anderen Programmiersprache schon gesehen. Der Unterschied, und damit auch der Teufel, steckt - wie immer - im Detail. Sollten Sie sich bei der Programmierung nicht so ganz sicher sein, werfen Sie immer wieder einen Blick auf Tabelle 19.3, und sehen Sie nach, welchen Operator Sie bei Ihren Berechnungen benötigen!

Operator	Beschreibung	Beispiel
+	Addition	11 + 4 (ergibt 15)
-	Subtraktion	11 - 4 (ergibt 7)
*	Multiplikation	11 * 4 (ergibt 44)
/	Division	11 / 4 (ergibt 2.75)
%	Rest bei Division	11 % 4 (ergibt 3)

**Tabelle 19.3: Die arithmetischen Operatoren von PHP**

Zusammen mit dem Vergleichsoperator können Sie nun schon einige Dinge mit Zahlen anstellen:

```
<?php
 $AnzahlWochen = 52;
 $TageProWoche = 7;
 $TageProJahr = 2 + $AnzahlWochen * $TageProWoche;
 print "Ein Schaltjahr hat $TageProJahr Tage";
?>
```



Sie sehen am Ergebnis (366): Es gilt Punkt-vor-Strich. Die Multiplikation wird also vor der Addition ausgeführt.

Natürlich gibt es bei diesen Operatoren auch wieder die schon aus JavaScript und Perl bekannten Kurzformen:

Kurzform	Ausführliche Schreibweise	Beschreibung
<code>\$x++;</code>	<code>\$x = \$x + 1;</code>	Inkrement
<code>\$x--;</code>	<code>\$x = \$x - 1;</code>	Dekrement
<code>\$x += \$y;</code>	<code>\$x = \$x + \$y;</code>	Addition
<code>\$x -= \$y;</code>	<code>\$x = \$x - \$y;</code>	Subtraktion
<code>\$x *= \$y;</code>	<code>\$x = \$x * \$y;</code>	Multiplikation
<code>\$x /= \$y;</code>	<code>\$x = \$x / \$y;</code>	Division
<code>\$x %= \$y;</code>	<code>\$x = \$x % \$y;</code>	Rest bei Divison (Modulo)

**Tabelle 19.4: Die Kurzformen der arithmetischen Operatoren von PHP**

## Vergleichsoperatoren

Ein Vergleich macht meistens nur bei numerischen Werten Sinn, und unter Umständen noch bei Zeichenketten; dort spricht man von einem *lexikalischen Vergleich*. Ein 'A' ist damit »kleiner« als ein 'B', aufgrund der Position im Alphabet. Das Ergebnis eines Vergleichs ist eine boolesche Variable, also wahr (true) oder falsch (false). PHP kennt die in Tabelle 19.5 aufgeführten Vergleichsoperatoren.

Operator	Beschreibung	Beispiel
<code>&gt;</code>	größer als	<code>2 &gt; 1</code> (ergibt true)
<code>&lt;</code>	kleiner als	<code>2 &lt; 1</code> (ergibt false)
<code>&gt;=</code>	größer oder gleich als	<code>2 &gt;= 1</code> (ergibt true)
<code>&lt;=</code>	kleiner oder gleich als	<code>2 &lt;= 1</code> (ergibt false)
<code>==</code>	Gleich	<code>2 == 1</code> (ergibt false)
<code>&lt;&gt;</code>	Ungleich	<code>2 &lt;&gt; 1</code> (ergibt true)

## Tabelle 19.5: Die Vergleichsoperatoren von PHP



*Zwei mögliche Fehlerquellen gilt es zu beachten. Erstens, Gleichheit wird durch ein doppeltes Gleichheitszeichen überprüft, nicht durch ein einfaches, das ist eine Zuweisung! Und zweitens, es gibt (im Gegensatz zu Perl) keine eigenen Vergleichsoperatoren für Strings, Sie können also bei PHP Strings mit den Operatoren aus Tabelle 19.5 vergleichen.*

## Logische Operatoren

Auch mit booleschen Variablen, oder Wahrheitswerten, lassen sich Operationen durchführen. Zuständig hierfür sind die logischen Operatoren, welche Sie in Tabelle 19.6 aufgeführt finden.

Operator	Bedeutung	Beispiel
and	(logisches) Und	true and false (ergibt false)
&&	(logisches) Und	true && false (ergibt false)
or	(logisches) Oder	true or false (ergibt true)
	(logisches) Oder	true    false (ergibt true)
!	Nicht (Negation)	not true (ergibt false)
xor	Entweder oder	true xor false (ergibt true)

## Tabelle 19.6: Die logischen Operatoren von PHP



*Wie Sie sehen, sind and und && bzw. or und || ebenbürtig. In der Praxis verwendet man jedoch fast ausschließlich && und ||.*

## Zeichenkettenoperatoren und -operationen

Mit Zeichenketten können Sie zwar nicht groß rechnen, aber zumindest werden Sie sie hin und wieder konkatenieren (aneinanderhängen) müssen. Als Operator dient hierzu *nicht* das Plus-Zeichen, sondern - wie in Perl auch - der Punkt. Um eines der ersten Beispiele des heutigen Tages, die Ausgabe einer Handvoll von Variablen, neu aufzuwärmen und etwas kürzer zu fassen, können wir uns dieses Operators bedienen.

## Listing 19.6: konkatenation.php

```
<?php
 $begrueßung = "Guten Tag!";
 $tag = 18;
 $thema = "PHP";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Konkatenation</title>
</head>
<body background="white">
<?php
 print "<h1>".$begrueßung."</h1>\n";
 print "Herzlich Willkommen an Tag ".$tag."; \n";
 print "das heutige Thema lautet: ".$thema.". \n";
?>
</body>
</html>
```



*Natürlich hätten wir hier auch Interpolation einsetzen können.*



*Auch für diesen Operator gibt es eine Kurzform:  $\$x .= \$y$ ; entspricht  $\$x = \$x . \$y$ ; - an die Variable  $\$x$  wird also der Wert der Variablen  $\$y$  angehängt.*

Ansonsten kennt PHP noch eine Reihe von String-Funktionen; eine komplette Übersicht finden Sie unter der URL [www.php.net/manual/ref.strings.php](http://www.php.net/manual/ref.strings.php). Wir beschränken uns an dieser Stelle auf die wichtigsten Funktionen (wobei wir natürlich eine subjektive Auswahl treffen mussten). In Tabelle 19.7 finden Sie außer dieser Übersicht außerdem noch die entsprechende äquivalente VBScript-Funktion, so dass Sie einen Vergleich haben und Skripte schneller von der einen in die andere Sprache portieren können:

Funktion	Beschreibung	Parameter	VBScript-Äquivalent
----------	--------------	-----------	---------------------

strlen(s)	Ermittelt die Länge eines Strings	s - der betrachtete String	Len(s)
strpos(s1, s2)	Ermittelt die <i>erste</i> Position eines Teilstrings in einem längeren String, wobei die Zählung bei 0 beginnt. Der Rückgabewert -1 bedeutet, dass der Teilstring nicht im längeren String enthalten war.	s1 - der zu durchsuchende String  s2 - der zu suchende String	InStr(s1, s2)
strrpos(s1, s2)	Ermittelt die <i>letzte</i> Position eines Teilstrings in einem längeren String, wobei die Zählung bei 0 beginnt. Der Rückgabewert -1 bedeutet, dass der Teilstring nicht im längeren String enthalten war.	s1 - der zu durchsuchende String  s2 - der zu suchende String	InStrRev(s1, s2)
substr(s, a, b)	Liefert einen Teilstring aus einem String zurück, wobei die Zählung bei 0 beginnt.	s - der betrachtete String  a - Startposition des Teilstrings  b - Anzahl der zu ermittelnden Zeichen	Mid(s, a, b)
str_replace(s1, s2, s)	Ersetzt in einem String einen Teilstring durch einen anderen und liefert den ersetzten String zurück	s1 - der zu ersetzende Teilstring  s2 - der Teilstring, durch den ersetzt werden soll  s - der betrachtete String	Replace(s, s1, s2)

**Tabelle 19.7: Einige der Zeichenkettenfunktionen von PHP**

Im Folgenden ein kleines Beispiel, das den Einsatz dieser Funktionen demonstriert:

**Listing 19.7: zeichenketten.php**

```
<?php
```

```
$s = "Markt und Technik";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Zeichenkettenfunktionen</title>
</head>
<body bgcolor="white">
<p>

 strlen:
 <?php print strlen($s); ?>

 strpos:
 <?php print strpos($s, "n"); ?>

 strrpos:
 <?php print strrpos($s, "n"); ?>

 substr:
 <?php print substr($s, 6, 3); ?>

 str_replace:
 <?php print str_replace("und", "+", $s); ?>

 $s:
 <?php print $s; ?>

</p>
</body>
</html>
```

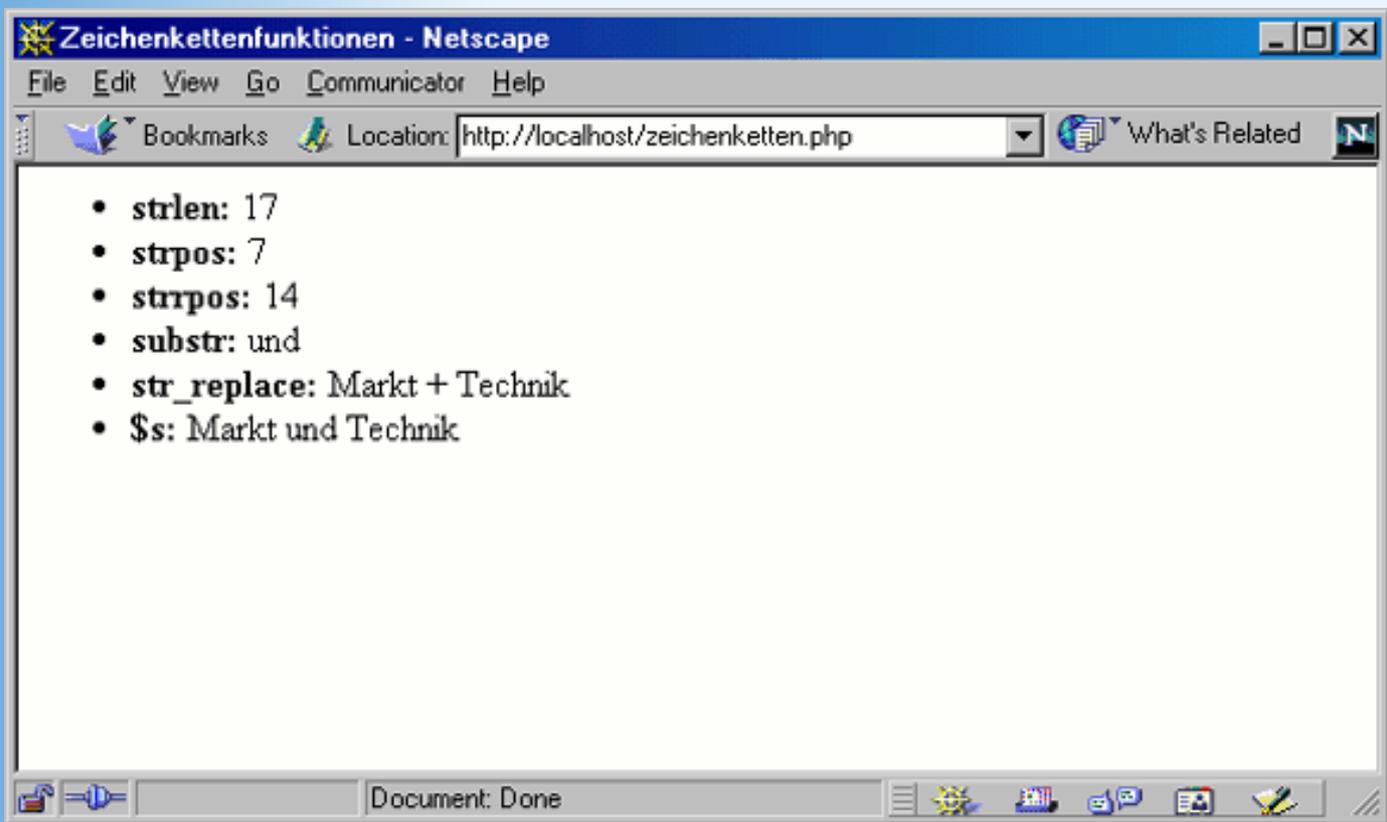


Abbildung 19.5: Die Zeichenkettenfunktionen im Einsatz



*Beachten Sie, dass sich - wie in der letzten Ausgabezeile in Abbildung 19.5 zu sehen - der Wert der Variablen `$s` selbst nie ändert, auch durch die Anwendung von `str_replace` nicht!*

## Fallunterscheidungen

Egal, welcher Programmiersprache man sich zuwendet: Fast alle haben irgendeine Form von Fallunterscheidung, und bei fast allen gibt es `if` - so auch in PHP. In Sachen Syntax herrscht fast völlige Gleichheit zu JavaScript und Perl; ein Grund mehr, dieses Thema hier recht kurz abzuhandeln!

Damit wir an dieser Stelle ein halbwegs realistisches Beispiel anbringen können, hier ein kleiner Vorgriff auf später: Die PHP-Funktion `getdate` liefert ein assoziatives Array zurück. Über den Schlüssel `wday` erhalten Sie den aktuellen Wochentag als numerischen Wert. 0 steht dabei für Sonntag, 1 für Montag, und so weiter, bis zum Wert 6, der für Samstag steht. Das funktioniert also ganz analog zur `Weekday`-Funktion von VBScript, nur das letztere einen um 1 höheren Wert für jeden Tag zurückliefert.

```
<?php
 $datum = getdate();
 print $datum["wday"];
```

?>

## if-Anweisung

Nun aber zur if-Anweisung selbst. In der kürzesten Form sieht sie folgendermaßen aus:

```
<?php
 if (Bedingung) {
 Befehlsblock
 }
?>
```

## else-Anweisung

Natürlich gibt es in PHP auch else, das wie folgt eingesetzt wird:

```
<?php
 if (Bedingung) {
 Befehlsblock
 } else {
 Befehlsblock
 }
?>
```

Um das Wochentagsbeispiel von gestern noch einmal aufzuwärmen:

```
<?php
 $datum = getdate();
 $tag = $datum["wday"];
 if ($tag == 0 || $tag == 6) {
 print "Am Wochenende wird nicht gearbeitet!";
 } else {
 print "Guten Tag, frisch ans Werk!";
 }
?>
```

## elseif-Anweisung

Sie wissen bereits, dass viele else-Anweisungen zu unübersichtlichem Code führen (können). Aus diesem Grund gibt es auch in PHP elseif, welches wie folgt eingesetzt wird:

```
<?php
 if (Bedingung) {
 Befehlsblock
 } elseif (Bedingung) {
 Befehlsblock
 } elseif (Bedingung) {
 ...
 }
```

```
} else {
 Befehlsblock
}
?>
```

Die letzte else-Anweisung samt folgendem Befehlsblock ist dabei optional.

Zurück zum Wochentagsbeispiel, diesmal ausführlicher und mit elseif:

```
<?php
$datum = getdate();
$tag = $datum["wday"];
if ($tag == 0 || $tag == 6) {
 print "Am Wochenende wird nicht gearbeitet!";
} elseif ($tag == 3 || $tag == 4 || $tag == 5) {
 print "Guten Tag, frisch ans Werk!";
} else {
 print "So früh in der Woche, und schon surfen?!";
}
?>
```

## switch-Anweisung

Was in VBScript Select Case heißt, heißt in PHP (und auch in JavaScript) switch. Hiermit können Sie eine Variable/einen Ausdruck gegen mehrere mögliche Werte prüfen. Die Syntax ist die folgende:

```
<?php
switch(Variable) {
 case Wert:
 Befehlsblock
 break;
 case Wert:
 Befehlsblock
 break;
 ...
 default:
 Befehlsblock
}
?>
```

Der Block mit default und dazugehörigem Befehlsblock ist optional. Hier können Sie die Standardbefehle angeben, die abgearbeitet werden sollen, falls die Variable mit keinem der angegebenen Werte übereinstimmt.

Die Anweisung break ist ebenfalls sehr wichtig. Hierdurch wird die switch-Anweisung sofort verlassen, nachdem der entsprechende Befehlsblock abgearbeitet worden ist. Andernfalls würden alle Befehlsblöcke abgearbeitet werden, sobald die Variable mit einem der Werte

übereinstimmt. Der PHP-Interpreter geht eine switch-Anweisung zeilenweise durch.

Hiermit kann das Beispiel mit dem Wochentag umgeschrieben werden. Im Gegensatz zu VBScript unterstützt switch jedoch nicht die Angabe von mehreren Werten innerhalb einer einzelnen case-Anweisung! Sie müssen also jeden Wert einzeln angeben.

```
<?php
 $datum = getdate();
 $tag = $datum["wday"];
 switch ($tag){
 case 0: print "Sonntag, Zeit zu rasten"; break;
 case 1: print "Montag, Arbeitsbeginn"; break;
 case 2: print "Dienstag, immer noch Arbeit"; break;
 case 3: print "Mittwoch, Wochenmitte"; break;
 case 4: print "Donnerstag, Woche fast rum"; break;
 case 5: print "Freitag, letzter Arbeitstag"; break;
 default: print "Samstag, endlich Wochenende";
 }
?>
```

## Mischung mit HTML-Code

Alle bis dato vorgestellten Fallunterscheidungen haben einen kleinen, aber feinen Nachteil. Die jeweiligen Befehlsblöcke können nur PHP-Anweisungen enthalten, aber keinen HTML-Code. Wenn Sie beides mischen wollen, müssen Sie eine spezielle Version dieser Anweisungen einsetzen. Dazu gibt es zwei grobe Grundregeln: Zum einen muss die Anweisung (z.B. if, else) mit einem Doppelpunkt enden, und zum andern gibt es eine besondere Anweisung am Ende, die mit end beginnt (z.B. endif). Die geschweiften Klammern entfallen dafür.

Werfen wir einmal einem Blick hierauf, und beginnen bei der if-Anweisung:

```
<?php
 $datum = getdate();
 $tag = $datum["wday"];
 if ($tag == 0 || $tag == 6) :
?>
Am Wochenende wird nicht gearbeitet!
<?php
 elseif ($tag == 3 || $tag == 4 || $tag == 5) :
?>
Guten Tag, frisch ans Werk!
<?php
 else :
?>
 So früh in der Woche, und schon surfen?!
<?php
 endif;
?>
```

Nach `if` (*Bedingung*), `elseif` (*Bedingung*) und `else` kommt ein Doppelpunkt. Die Anweisung wird mit `endif` beendet. Dazwischen ist jeder HTML-Code erlaubt.

Auch die `switch`-Anweisung hat diese alternative Form:

```
<?php
 $datum = getdate();
 $tag = $datum["wday"];
 switch ($tag) :
 case 0: ?> Sonntag, Zeit zu rasten <?php break;
 case 1: ?> Montag, Arbeitsbeginn <?php break;
 case 2: ?> Dienstag, immer noch Arbeit <?php break;
 case 3: ?> Mittwoch, Wochenmitte <?php break;
 case 4: ?> Donnerstag, Woche fast rum <?php break;
 case 5: ?> Freitag, letzter Arbeitstag <?php break;
 default: ?> Samstag, endlich Wochenende <?php
 endswitch;
?>
```



Weitere Informationen über die alternative Syntax finden Sie unter [www.php.net/manual/control-structures.alternative-syntax.php](http://www.php.net/manual/control-structures.alternative-syntax.php).

## Schleifen

Wie Sie bereits aus den vorhergehenden Kapiteln wissen, dienen Schleifen zu weitaus mehr als zur reinen Tippersparnis bei der Programmierung. Auch PHP bietet eine Reihe von Schleifen an, und die gute Nachricht: Als JavaScript- und Perl-Experte kennen Sie alle. Aus diesem Grund können wir uns hier wieder ziemlich kurz fassen. Und noch eine gute Nachricht: Nach diesem Abschnitt lassen wir die Theorie hinter uns und steigen endlich in die Praxis ein!

### for-Schleife

Die `for`-Schleife dient primär zur wiederholten Ausführung eines Befehlsblocks. Dabei werden drei Parameter angegeben: Eine Start-Anweisung, eine Bedingung, die vor jeder Ausführung der Schleife überprüft wird, sowie eine Anweisung, die nach jedem Schleifendurchlauf ausgeführt wird. Folgender Code beispielsweise wiederholt die Argumentationsanregung zur Gehaltsverhandlung vom gestrigen Tage und gibt die Geldforderung zehnmal aus:

#### Listing 19.8: `gehalt1.php`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
```

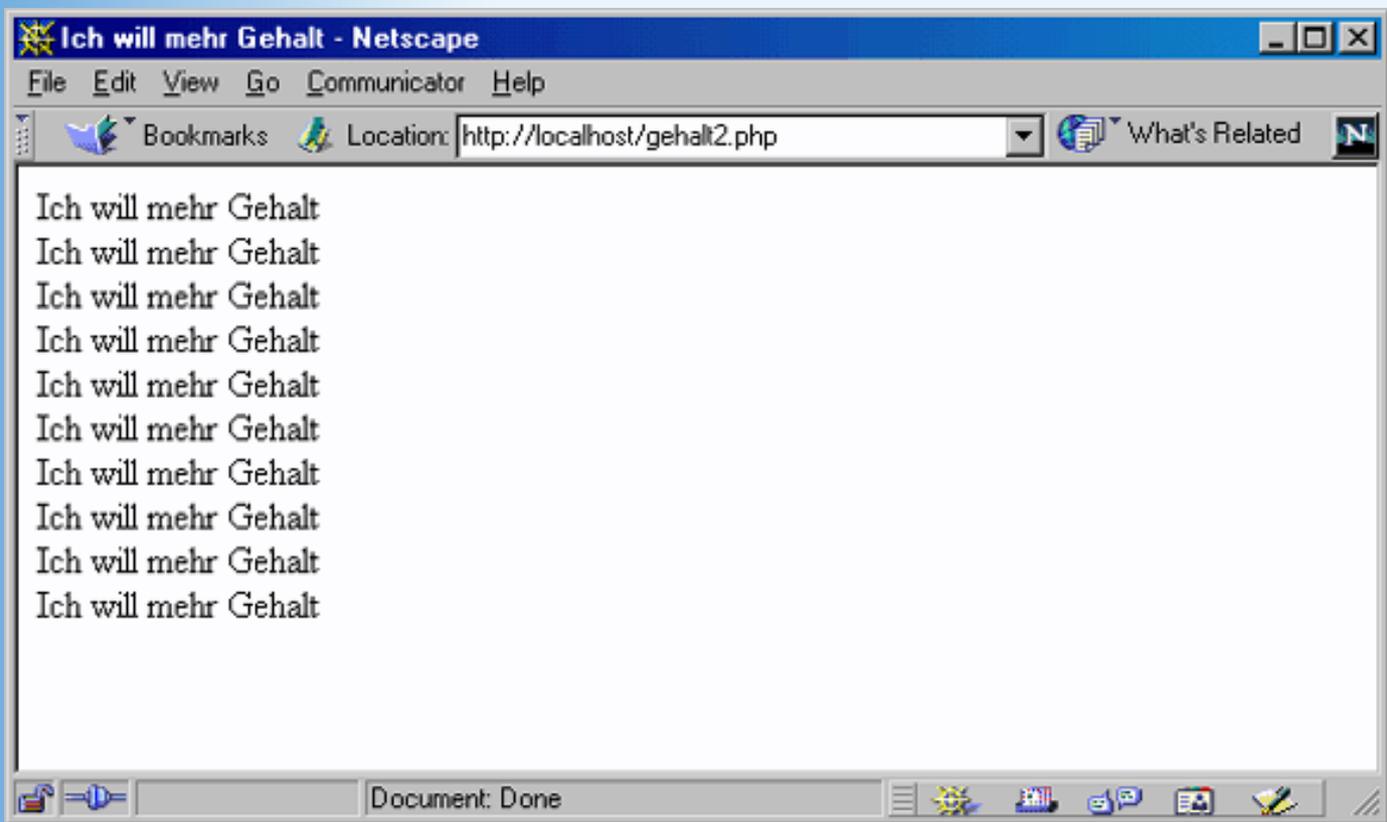
```
<head>
<title>Ich will mehr Gehalt</title>
</head>
<body bgcolor="white">
<p>
<?php
 for ($i = 1; $i <= 10; $i++) {
 print "Ich will mehr Gehalt
";
 }
?>
</p>
</body>
</html>
```

Auch hier gibt es wieder eine alternative Syntax; Weitere Informationen darüber finden Sie unter der bereits einmal aufgeführten URL [www.php.net/manual/control-structures.alternative-syntax.php](http://www.php.net/manual/control-structures.alternative-syntax.php), oder werfen Sie ein Beispiel auf folgenden Code:

### Listing 19.9: gehalt2.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Ich will mehr Gehalt</title>
</head>
<body bgcolor="white">
<p>
<?php
 for ($i = 1; $i <= 10; $i++) :
?>
 Ich will mehr Gehalt

<?php
 endfor;
?>
</p>
</body>
</html>
```



**Abbildung 19.6: Gehaltsverhandlungen mit PHP - ob's funktioniert?**

In der Praxis wird die Zählvariable, die im ersten Teil der Klammer nach for initialisiert wird, innerhalb der Schleife verwendet. Beispielsweise gibt folgendes Skript die Buchstaben einer Zeichenkette einzeln aus und verwendet dabei eine Schleife:

#### **Listing 19.10: pappenheimer.php**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Pappenheimer</title>
</head>
<body bgcolor="white">
<p>
<?php
 $p = "Pappenheimer";
 for ($i = 0; $i < strlen($p); $i++) :
?>
 Das <?=$i ?>. Zeichen ist: <?=substr($p, $i, 1) ?>

<?php
 endfor;
?>
</p>
</body>
</html>
```

## while-Schleife

Bei der while-Schleife wird am Anfang jedes Schleifendurchlaufs eine Bedingung überprüft. Solange diese Bedingung erfüllt ist, wird der Anweisungsblock innerhalb der Schleife ausgeführt:

### Listing 19.11: gehalt3.php

```
<?php
 $i = 0;
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Ich will mehr Gehalt</title>
</head>
<body bgcolor="white">
<p>
<?php
 while ($i < 10) {
 print "Ich will mehr Gehalt
";
 $i++;
 }
?>
</p>
</body>
</html>
```

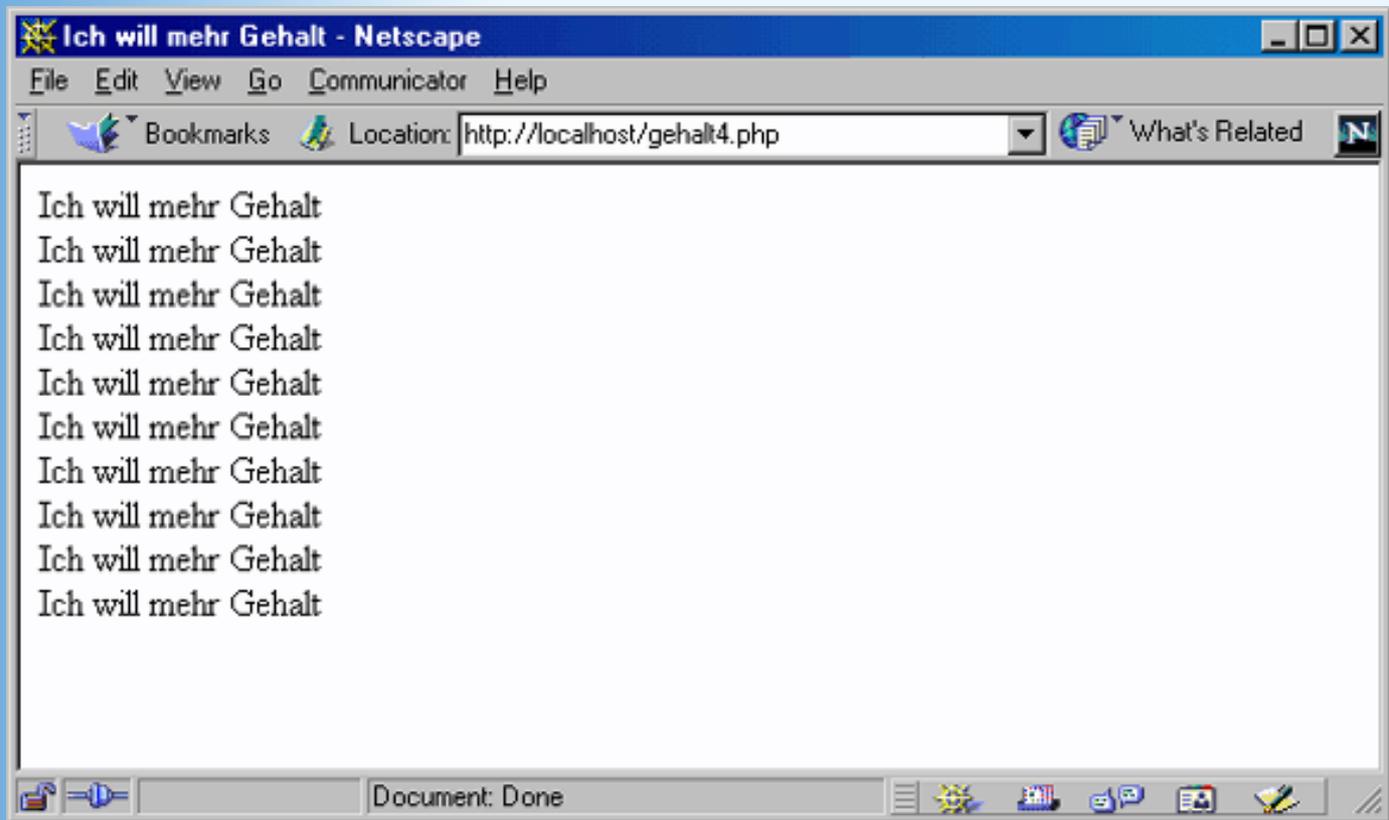
Auch von dieser Schleife gibt es eine alternative Syntax:

### Listing 19.12: gehalt4.php

```
<?php
 $i = 0;
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Ich will mehr Gehalt</title>
</head>
<body bgcolor="white">
<p>
<?php
 while ($i < 10) :
?>
Ich will mehr Gehalt

<?php
```

```
$i++;
endwhile;
?>
</p>
</body>
</html>
```



**Abbildung 19.7: Mehr Gehalt - dank while/endwhile**

## do-while-Schleife

Die do-while-Schleife ähnelt der while-Schleife sehr. Es gibt nur zwei augenscheinliche Unterschiede. Zunächst einmal wird die Bedingung erst nach Ausführung des Anweisungsblocks überprüft (die Schleife wird also mindestens einmal durchlaufen), und zweitens gibt es keine alternative Syntax. Ansonsten bringt diese Schleife aber auch nicht viel Neues.

### Listing 19.13: gehalt5.php

```
<?php
 $i = 0;
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Ich will mehr Gehalt</title>
</head>
<body bgcolor="white">
```

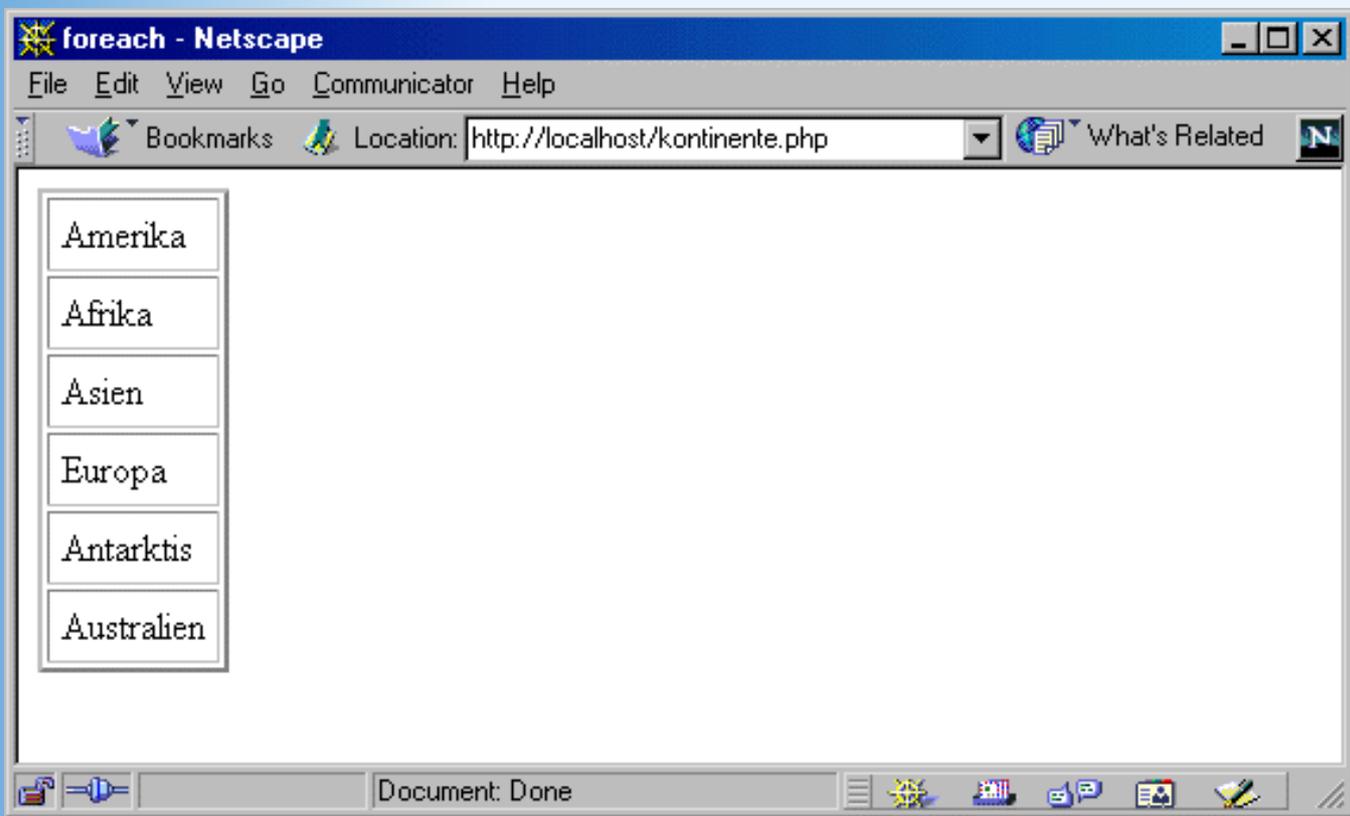
```
<p>
<?php
 do {
 print "Ich will mehr Gehalt
";
 $i++;
 } while ($i < 10)
?>
</p>
</body>
</html>
```

## foreach-Schleife

Sie kennen foreach-Konstrukte aus JavaScript (heißt dort for in), ASP (For Each) und Perl, und deswegen haben Sie wohl schon eine Vermutung, wozu diese Schleife eingesetzt wird: Zum Durchschreiten einer Auflistung oder einer Kollektion. Beispielsweise können Sie auf alle Elemente eines Arrays zugreifen. Diese Funktion wurde übrigens erst in PHP-Version 4 eingeführt, unter PHP 3 war dies noch völlig unbekannt.

Zunächst einmal können Sie alle Werte eines Arrays mit einer Schleife und wenig zusätzlichem Aufwand ausgeben:

```
<?php
 $kontinente = array("Amerika", "Afrika", "Asien", "Europa",
 "Antarktis", "Australien");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>foreach</title>
</head>
<body bgcolor="white">
<table border="2" cellpadding="5">
<?php
 foreach ($kontinente as $k) {
 print "<tr><td>$k</td></tr>";
 }
?>
</table>
</body>
</html>
```

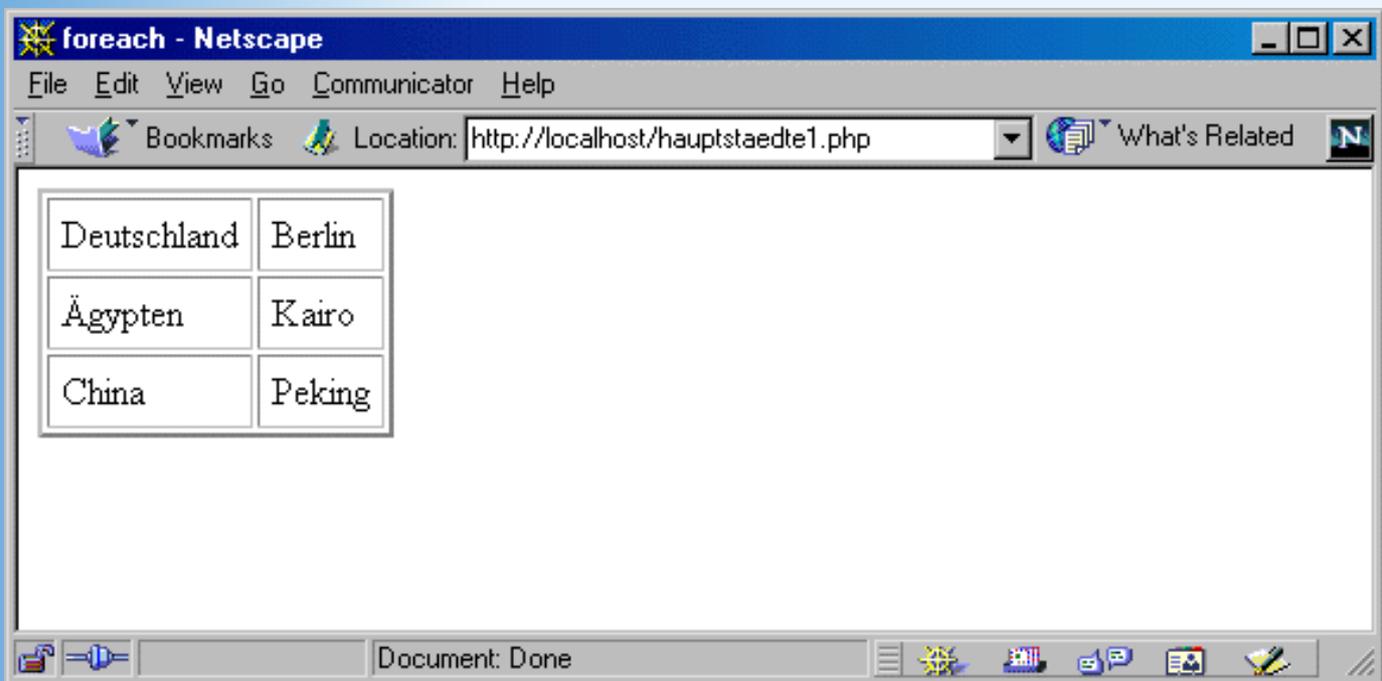


**Abbildung 19.8: Die sechs Kontinente - ausgegeben mit foreach**

Die zweite Variante der foreach-Schleife greift bei assoziativen Arrays: Sie erhalten die einzelnen Schlüssel des Arrays und die Werte dazu:

#### **Listing 19.14: hauptstaedte1.php**

```
<?php
 $hauptstadt = array("Deutschland"=>"Berlin", "Ägypten"=>"Kairo",
 "China"=>"Peking");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>foreach</title>
</head>
<body bgcolor="white">
<table border="2" cellpadding="5">
<?php
 foreach ($hauptstadt as $land => $stadt) {
 print "<tr><td>$land</td><td>$stadt</td></tr>";
 }
?>
</table>
</body>
</html>
```



**Abbildung 19.9:** Auch assoziative Arrays können mit foreach ausgegeben werden

Auch von dieser, letzten, Schleife gibt es eine alternative Form. Wie üblich: Die foreach-Anweisung endet mit einem Doppelpunkt, und am Ende des Anweisungsblocks kommt endforeach.

### Listing 19.15: hauptstaedte2.php

```
<?php
 $hauptstadt = array("Deutschland"=>"Berlin", "Ägypten"=>"Kairo",
 "China"=>"Peking");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>foreach</title>
</head>
<body bgcolor="white">
<table border="2" cellpadding="5">
<?php
 foreach ($hauptstadt as $land => $stadt) :
?>
<tr><td><?=$land ?></td><td><?=$stadt ?></td></tr>
<?php
 endforeach;
?>
</table>
</body>
</html>
```

Damit sind wir am Ende der PHP-Spracheinführung angelangt. Doch wir haben noch nicht das

Ende des heutigen Tags erreicht, denn jetzt steigen wir erst richtig in die Materie ein und führen einige Standardbeispiele vor. Schnallen Sie sich also an!

## 19.3 Datumswerte

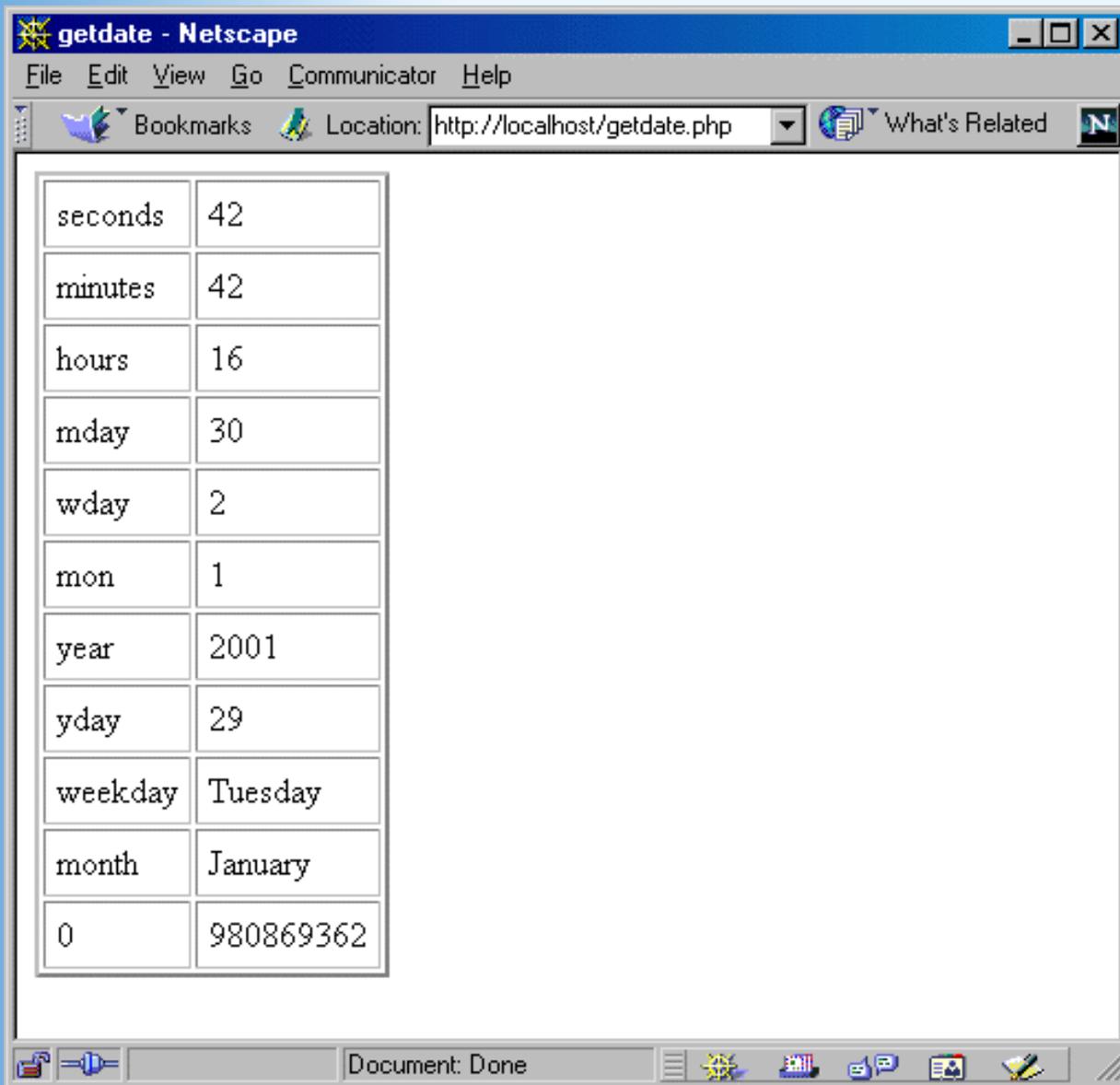
Es ist schon ein Unterschied, ob man serverseitig oder clientseitig programmiert. Nicht nur, dass serverseitige Programmierung ein wenig mehr Können voraussetzt - die Anforderungen an den Server sind ungleich größer. Aus Sicherheitsbedenken, oder auch aus Bequemlichkeit, ist bei vielen Hosting-Paketen keine Unterstützung von serverseitigen Technologien dabei. Oder, aus der anderen Sicht betrachtet: Bei Billigangeboten rechnet sich für den Hostler der zusätzliche administrative Aufwand einfach nicht.

Wenn es aber soweit ist, sprich, man hat serverseitige Mittel wie eben PHP zur Verfügung, fängt das große Kopfkratzen oft erst an. Was will ich überhaupt serverseitig? Und vor allem, was lässt sich vergleichsweise schnell und unauffällig umsetzen? Die Antwort lautet sehr oft: Gib zumindest einmal das aktuelle Tagesdatum aus. Klar, das geht auch mit JavaScript, aber das setzt eben gewisse Anforderungen an den Client (JavaScript muss aktiviert sein, Browser muss eine bestimmte Versionsnummer haben), und darüber haben Sie keine Kontrolle. Sie haben aber Kontrolle über den Server (hoffentlich).

Sie sind heute schon einmal mit der Funktion `getdate` in Berührung gekommen. Diese gibt ein assoziatives Array zurück, über das sie einzelne Informationen über das aktuelle Tagesdatum (beispielsweise Tag, Monat, Jahr) herausbekommen können. Werfen wir zunächst einen Blick darauf, was genau in diesem Array alles steht. Hierbei hilft uns `foreach`:

### Listing 19.16: `getdate.php`

```
<?php
 $datum = getdate();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>getdate</title>
</head>
<body bgcolor="white">
<table border="2" cellpadding="5">
<?php
 foreach ($datum as $schluessel => $wert) :
?>
<tr><td><?=$schluessel ?></td><td><?=$wert ?></td></tr>
<?php
 endforeach;
?>
</table>
</body>
</html>
```



**Abbildung 19.10: Alle Werte im von getdate zurückgelieferten Array**

Führen Sie obiges Skript einmal bei sich aus, und versuchen Sie dann, die Bedeutung der einzelnen Werte (bzw. deren Schlüssel) zu interpretieren. Fertig? Dann vergleichen Sie doch Ihre Ergebnisse mit der »offiziellen« Lösung in Tabelle 19.8.

Schlüssel	Beschreibung	Rückgabewert am 24.12.2001, 23:45:56 Uhr
Seconds	Sekunde	56
Minutes	Minute	45
Hours	Stunde	23
Mday	Tag	24
Wday	Wochentag (0=Sonntag, 6=Samstag)	1
Mon	Monat	12

Year	Jahr	2001
Yday	Tag im Jahr (fortlaufend gezählt)	358
Weekday	Wochentag als (englischer) Text	Monday
Month	Monat als (englischer) Text	December

**Tabelle 19.8: Datumsinformationen in getdate**



*Wenn Sie mehr über eine Funktion herausfinden wollen, hilft Ihnen das Online-Manual zu PHP weiter. Sie können direkt zu den meisten Funktionen springen, indem Sie die URL [www.php.net/Funktionsname](http://www.php.net/Funktionsname) aufrufen. Beispielsweise erfahren Sie mehr zu `getdate`, wenn Sie [www.php.net/getdate](http://www.php.net/getdate) aufrufen. Sie werden dann automatisch zur »richtigen« URL, in diesem Falle [www.php.net/manual/function.getdate.php](http://www.php.net/manual/function.getdate.php), weitergeleitet.*

Somit ist es nun ein Leichtes, ein schön formatiertes Datum auszugeben. Als kleines Extra übersetzen wir die Monats- und Tagesnamen vom Englischen ins Deutsche. Die erste Möglichkeit besteht darin, in einem normalen Array die numerischen Werte auf Strings abzubilden:

### Listing 19.17: datum.php

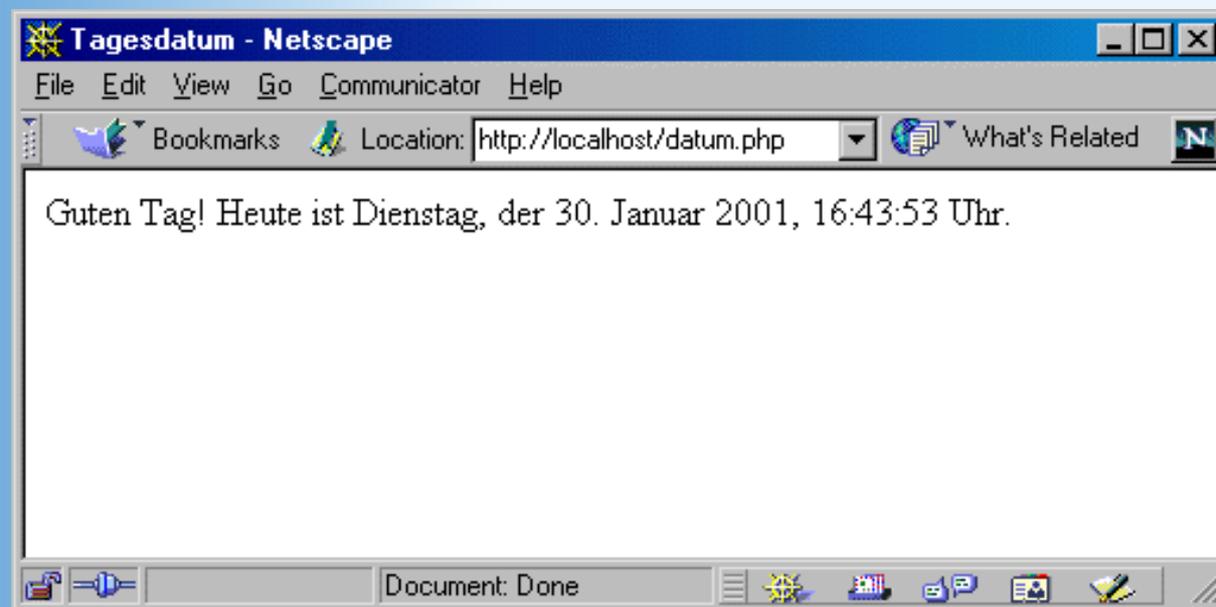
```
<?php
 $monatsnamen = array("", "Januar", "Februar", "Märzt",
 "April", "Mai", "Juni", "Juli", "August",
 "September", "Oktober", "November", "Dezember");
 $tagesnamen = array("Sonntag", "Montag", "Dienstag", "Mittwoch",
 "Donnerstag", "Freitag", "Samstag");

 $datum = getdate();
 $anzeige = $tagesnamen[$datum["wday"]];
 $anzeige .= ", der ";
 $anzeige .= $datum["mday"] . ". ";
 $anzeige .= $monatsnamen[$datum["mon"]] . " ";
 $anzeige .= $datum["year"] . ", ";
 $anzeige .= $datum["hours"] . ":";
 $anzeige .= $datum["minutes"] . ":";
 $anzeige .= $datum["seconds"] . " Uhr.";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
```

```

<title>Tagesdatum</title>
</head>
<body bgcolor="white">
<p>Guten Tag! Heute ist <?php print $anzeige; ?></p>
</body>
</html>

```



**Abbildung 19.11: Das aktuelle, formatierte Datum**

Alternativ können Sie auch ein assoziatives Array einsetzen, um die englischen auf die deutschen Bezeichnungen umzusetzen. Der Rest des Skripts funktioniert dann analog, Sie müssen nur bei Wochentag und Monat die richtigen Schlüssel für das Rückgabearray von `getdate` wählen.

#### Listing 19.18: `datum_assoziativ.php`

```

<?php
 $monatsnamen = array("January" => "Januar", "February" => "Februar",
 "March" => "Märzt", "April" => "April",
 "May" => "Mai", "June" => "Juni",
 "July" => "Juli", "August" => "August",
 "September" => "September",
 "October" => "Oktober", "November" => "November",
 "December" => "Dezember");
 $tagesnamen = array("Sunday" => "Sonntag", "Monday" => "Montag",
 "Tuesday" => "Dienstag", "Wednesday" => "Mittwoch",
 "Thursday" => "Donnerstag", "Friday" => "Freitag",
 "Saturday" => "Samstag");
 $datum = getdate();
 $anzeige = $tagesnamen[$datum["weekday"]];
 $anzeige .= ", der ";
 $anzeige .= $datum["mday"] . ". ";
 $anzeige .= $monatsnamen[$datum["month"]] . " ";

```

```

$anzeige .= $datum["year"] . ", ";
$anzeige .= $datum["hours"] . ":";
$anzeige .= $datum["minutes"] . ":";
$anzeige .= $datum["seconds"] . " Uhr.";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Tagesdatum</title>
</head>
<body bgcolor="white">
<p>Guten Tag! Heute ist <?php print $anzeige; ?></p>
</body>
</html>

```

## 19.4 Auf Formulare zugreifen

Richtig interessant wird PHP aber erst, wenn Sie Formulareingaben verarbeiten können und somit Interaktion mit Ihren Benutzern erst ermöglicht wird. Bei PHP ist dies deutlich einfacher als bei JavaScript, Perl und ASP, zumindest war es das. Es gibt aber Bestrebungen, den Aufwand für den Programmierer wieder ein wenig zu vergrößern. Warum dies so ist, erfahren Sie in den folgenden Abschnitten.

### Allgemeines

Unter Version 3 von PHP war das Formularhandling sehr, sehr bequem, und auch besonders leicht zu erlernen. Nehmen wir einmal an, Sie haben ein Formularelement mit name-Attribut »Pappenheimer«. Dann können Sie wie folgt auf das zugehörige value- Attribut zugreifen: \$Pappenheimer.

Sie haben richtig gelesen. Das name-Attribut ist gleichzeitig Namensgeber für die dazugehörige Variable. Sie müssen nun zwar für die name-Attribute einige gesonderte Regeln beachten, beispielsweise keine Leer-, Sonderzeichen oder Bindestriche, aber ansonsten funktioniert das ganz tadellos. Sehen Sie selbst:

#### Listing 19.19: post.php

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Formulare</title>
</head>
<body bgcolor="white">
<h3>Formularwerte</h3>
<p>
<?php

```

```
print "Vorname: $vorname
";
print "Nachname: $nachname
";
print "E-Mail: $email
";
```

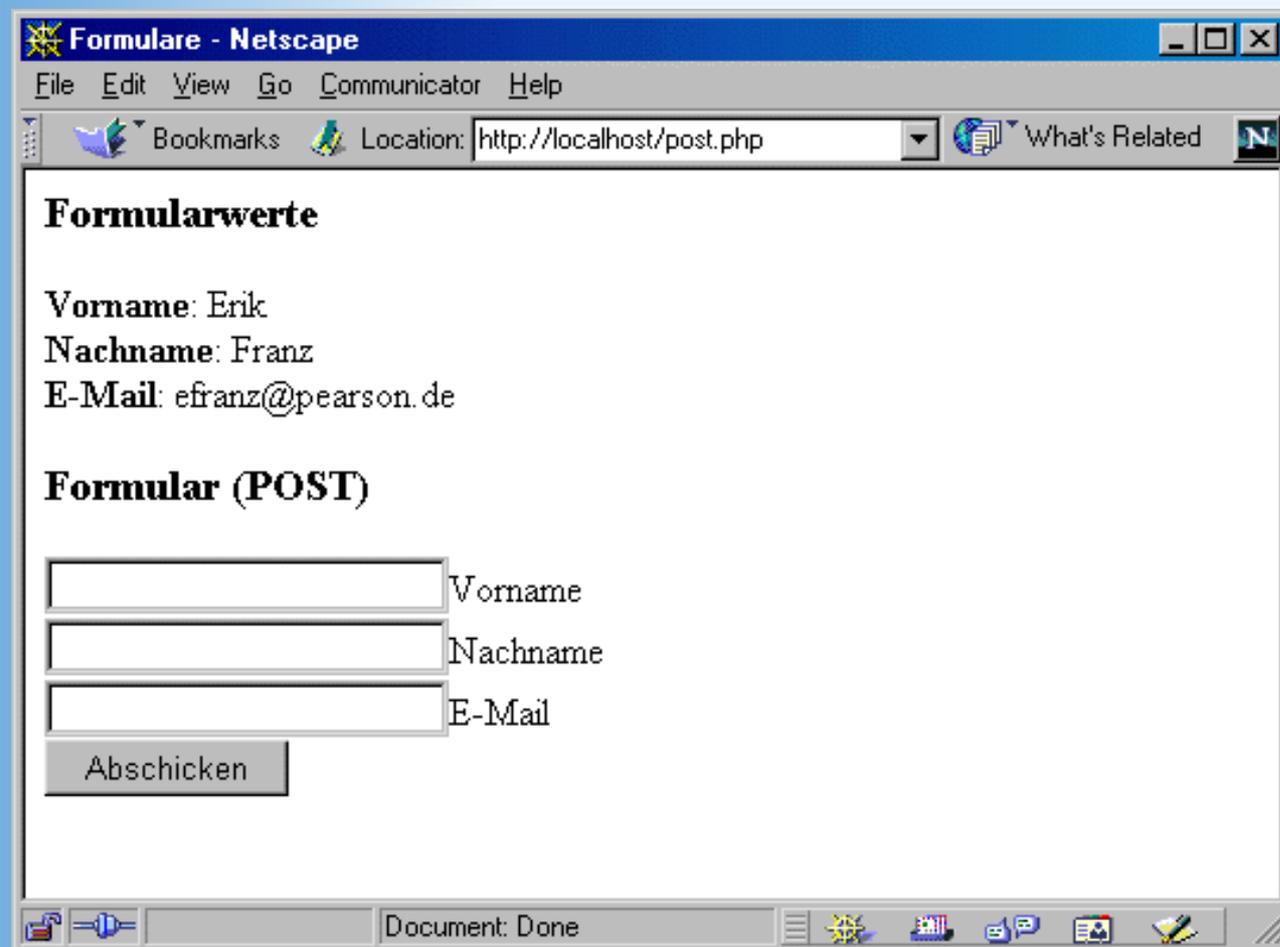
Next

```
?>
</p>
<h3>Formular (POST)</h3>
<p>
<form method="post">
<input type="text" name="vorname" size="20" />Vorname

<input type="text" name="nachname" size="20" />Nachname

<input type="text" name="email" size="20" />E-Mail

<input type="submit" name="submit" value="Abschicken" />
</form>
</p>
</body>
</html>
```



**Abbildung 19.12: Formulareingaben können mit PHP einfach erfasst werden**

Wie Sie auch aus Abbildung 19.12 entnehmen können, funktioniert obiger Code entsprechend den analogen ASP-Beispielen. Das action-Attribut des Formulars bleibt leer, dadurch wird das Formular auf das aktuelle Skript verschickt.



In der Variablen `$PHP_SELF` steht die URL der aktuellen Seite, Sie könnten also auch `action="<?php print $PHP_SELF; ?>"` im `<form>`-Tag verwenden. Diesen Tipp finden Sie allerorten, aber hier ist er - wie oben ausgeführt - schlichtweg unnötig!

So weit, so gut. Das mit den bequemen Variablennamen geht so weit, dass Sie auch auf Cookies auf dieselbe Art und Weise zugreifen können. Dies erinnert also entfernt an das Objekt Request vom gestrigen Tag (ASP): auch hier konnten Sie auf alle Formularvariablen zugreifen (egal ob das Formular via POST oder GET verschickt worden ist), sowie auf Cookies.

Der Nachteil an der ganzen Sache war und ist jedoch die Manipulierbarkeit. Stellen Sie sich vor, Sie programmieren einen geschützten Bereich mit PHP, und übergeben zur Authentifizierung der Benutzer irgendwelche geheimen Daten (seinen Benutzernamen, seine Kundennummer) als verstecktes Formularfeld oder als Cookie. Ein Wort vorweg: Mit etwas Aufwand kann der Benutzer auch selbst lokale Cookies anlegen. Sie fragen nun mit PHP den Wert im Cookie ab und gewähren bei Übereinstimmung mit einem bestimmten Wert den Zutritt zum geschützten Bereich. So weit, so gut, aber da Sie Cookies genauso wie POST-Daten und ebenfalls genauso wie GET-Variablen abfragen können, lässt sich dieser Mechanismus ganz einfach austricksen: Der Hacker übergibt die Daten einfach als Teil der URL, also via GET. Er ruft also nicht beispielsweise `login.php` auf, sondern `login.php?benutzer=erik`. Und schon nimmt Ihr Skript an, der Benutzer hat ein Cookie namens »benutzer« bei sich gespeichert, das den Wert »erik« hat.

In PHP 4 gibt es diese Vereinfachung immer noch. Allerdings gibt es in der `php.ini` jetzt besonders hervorgehoben einen Wert **register\_globals**. Dieser ist in der Standard- Version der `php.ini` auf **On** gesetzt, in der optimierten Version (`php.ini-optimized`) auf **Off**. Was bedeutet dies nun? Setzen Sie den Wert auf **On**, können Sie weiterhin über die kurzen Variablennamen auf die Formularwerte zugreifen. Ist **Off** eingestellt, so haben Sie nur noch die folgenden Möglichkeiten:

- Im assoziativen Array `$HTTP_POST_VARS` stehen die via POST verschickten Formulardaten
- Im assoziativen Array `$HTTP_GET_VARS` stehen die via GET verschickten Formulardaten

Inzwischen lautet die offizielle Empfehlung, die assoziativen Arrays zu verwenden. Auf fast jedem PHP-Kongress wird diese Forderung mehr oder weniger laut vorgebracht. Das Problem liegt jedoch darin, dass die meisten Anwendungen weltweit noch die »alte Methode« einsetzen, und es ist ein ziemlicher Aufwand, das Ganze umzuschreiben. So wird auch noch über längere Zeit hinweg eine Abwärtskompatibilität gegeben sein. Wenn Sie allerdings Anwendungen schreiben wollen, die auch noch in ferner Zukunft laufen, sollten Sie auf die Arrays zurückgreifen. Und auch, wenn **register\_globals** auf **On** gestellt ist, können Sie die Variablen dennoch verwenden. Wir werden uns deswegen heute an die Arrays halten.

## Unterschiede bei den einzelnen Formularelementen

Wie bei ASP gilt auch hier, dass es bei den einzelnen Typen von Formularfeldern Unterschiede in

der Handhabung innerhalb von PHP gibt. Dies gilt bei PHP umso stärker, wie Sie im Folgenden auch sehen werden. Ein Wort vorab noch: Wir verwenden als Versendemethode immer POST, und greifen dementsprechend auf `$HTTP_POST_VARS` zu. Natürlich können Sie dieselben Techniken mit GET einsetzen, Sie müssen dann lediglich an den entsprechenden Stellen mit `$HTTP_GET_VARS` arbeiten.

## Textfelder / Passwortfelder / unsichtbare Felder

Auch an dieser Stelle gibt es nichts Neues: `$name` liefert das value-Attribut des Elements mit name-Attribut name. Die folgenden Formularelemente sind hiervon betroffen:

- `<input type="text" />`
- `<input type="password" />`
- `<input type="hidden" />`
- `<textarea> ... </textarea>`

Beispiel:

Sie haben als E-Mail-Adresse

```
<?php print $HTTP_POST_VARS["email"]; ?>
```

angegeben!

## Auswahllisten

Bei normalen Auswahllisten gibt es auch hier kein Problem. Über `$name` erhalten Sie das value-Attribut des ausgewählten Elements der Auswahlliste mit name-Attribut name:

```
<p>Sie haben den folgenden Wert ausgew<hlt:

<?php print $name; ?></p>
```



*Wenn Sie keinen value-Wert bei einer Option der Auswahlliste angeben, verwenden die meisten Browser die Beschriftung der Option als value-Attribut - aber darauf sollten Sie sich nicht verlassen.*

Eine Besonderheit gibt es jedoch bei Mehrfach-Auswahllisten, die Sie mit `<select multiple>` erzeugen können. Damit Sie bei PHP auf alle ausgewählten Elemente zugreifen müssen, muss der HTML-Code angepasst werden. Das name-Attribut muss auf `[]` enden:

```
<select name="kontinente[]" multiple size="6">
 <option value="Amerika">Amerika</option>
 <option value="Afrika">Afrika</option>
 <option value="Asien">Asien</option>
 <option value="Europa">Europa</option>
```

```
<option value="Antarktis">Antarktis</option>
<option value="Australien">Australien</option>
</select>
```

Durch die eckigen Klammern wird dem PHP-Interpreter angezeigt, dass an dieser Stelle nicht nur ein Wert übergeben wird, sondern ein ganzes Array. Mit einer for-Schleife können Sie dann alle ausgewählten Werte ausgeben:

```
<p>
Sie haben die folgenden Kontinente bereits besucht:

<?php
 for ($i=0; $i<count($_HTTP_POST_VARS["kontinente"]); $i++) {
 print $_HTTP_POST_VARS["kontinente"][$i] . "
";
 }
</p>
```



Abbildung 19.13: Eine mögliche Ausgabe im Browser

## Checkboxes

Bei der Behandlung von Checkboxes tut man sich wiederum ein wenig leichter. Über das name-Attribut kommt man an das value-Attribut heran. Sie können nun einfach überprüfen, ob eine Checkbox angekreuzt worden ist oder nicht:

```
<?php
 if ($HTTP_POST_VARS["Name"] != "") {
 print "Checkbox angekreuzt!";
 } else {
 print "Checkbox nicht angekreuzt!";
 }
?>
```

Sie können alternativ auch noch die PHP-Funktion `isset` verwenden. Diese liefert `true` zurück, wenn eine Variable bereits mit einem Wert belegt worden ist, ansonsten `false`:

```
<?php
 if (isset($HTTP_POST_VARS["Name"])) {
 print "Checkbox angekreuzt!";
 } else {
 print "Checkbox nicht angekreuzt!";
 }
?>
```

## Radiobuttons

Da von einer Gruppe von Radiobuttons jeweils nur einer ausgewählt werden darf, gestaltet sich hier die Angelegenheit recht unkompliziert. Wie üblich: Über das `name`-Attribut der Gruppe erhält man das `value`-Attribut des angeklickten Radiobuttons.

Sie haben den Radiobutton

```
<?php print $HTTP_POST_VARS["Name"]; ?>
ausgewählt!
```

## Datei-Uploads

Kommen wir nun zu einem der großen Vorteile von PHP gegenüber ASP. Sie benötigen keine Zusatzkomponenten, um Datei-Uploads durchzuführen. Definieren Sie zunächst ein Datei-Upload-Formularelement in Ihrem HTML-Code:

```
<input type="file" name="datei" />
```



*Sie müssen als Versendemethode `POST` einstellen; ebenso müssen Sie `enctype="multipart/form-data"` setzen!*

Wenn Sie das Formular abschicken, passiert Folgendes:

- Der Browser überträgt die Datei an den Webserver.

- Der PHP-Interpreter speichert die Datei in einem temporären Verzeichnis ab, merkt sich aber, wo.
- Nachdem das PHP-Skript abgearbeitet worden ist, wird die temporäre Datei wieder gelöscht.

Bevor Sie mit diesem Formularelement herumexperimentieren, sollten Sie zunächst einen Blick auf Ihre *php.ini* werfen. Drei Einstellungen bedürfen einer besonderen Betrachtung:

- **file\_uploads**: Muss auf **On** gesetzt werden.
- **upload\_tmp\_dir**: Geben Sie hier ein temporäres Verzeichnis an. Falls Sie das nicht machen, nimmt das System den Standardwert, und unter Windows den Inhalt der Umgebungsvariablen **TEMP**. Überprüfen Sie also zumindest, ob diese Variable gesetzt ist, und ob jeder auf dieses Verzeichnis Schreibrechte hat.
- **upload\_max\_filesize**: Geben Sie hier den Maximalwert für die Größe der hochzuladenden Dateien an. Der Standardwert von 2 Megabyte ist beinahe schon zu hoch.

Nun können Sie loslegen. Da die temporäre Datei nach Abarbeitung des PHP-Skripts gelöscht wird, müssen Sie sie innerhalb des Skripts an eine andere Stelle kopieren, wenn Sie sie behalten wollen.

Zur Theorie: In obigem HTML-Beispiel hatte das Formularelement für den Datei-Upload das name-Attribut »datei«. Ihnen stehen dann (bei **register\_globals = On**) die folgenden Variablen zur Verfügung (nur unter PHP3!):

- \$datei - Name der temporären Datei (oder »none«, wenn keine Datei übertragen worden ist)
- \$datei\_name - Originaler Name der Datei, z.B. »grafik.jpg«
- \$datei\_size - Größe der Datei in Bytes
- \$datei\_type - MIME-Typ der Datei, z.B. »image/jpeg«

Aus oben ausgeführten Gründen sollten Sie jedoch wieder auf die entsprechenden assoziativen Arrays zugreifen - und hier müssen Sie es sogar (bei PHP 4). Für Datei-Uploads zuständig ist \$HTTP\_POST\_FILES. Ihnen stehen dann die folgenden Werte zur Verfügung:

- \$HTTP\_POST\_FILES["datei"]["tmp\_name"] - Name der temporären Datei (oder »none«, wenn keine Datei übertragen worden ist)
- \$HTTP\_POST\_FILES["datei"]["name"] - Originaler Name der Datei, z.B. »grafik.jpg«
- \$HTTP\_POST\_FILES["datei"]["size"] - Größe der Datei in Bytes
- \$HTTP\_POST\_FILES["datei"]["type"] - MIME-Typ der Datei, z.B. »image/jpeg«

Mit der Funktion `move_uploaded_file` können Sie dann die Datei an eine andere Position verschieben.

Folgender Code führt das einmal vor: Eine Datei wird zum Webserver übertragen, in das Hauptverzeichnis des Webserver kopiert, und dann angezeigt. Passen Sie das Hauptverzeichnis Ihres Webserver auf dem lokalen System entsprechend an.

**Listing 19.20: upload.php**

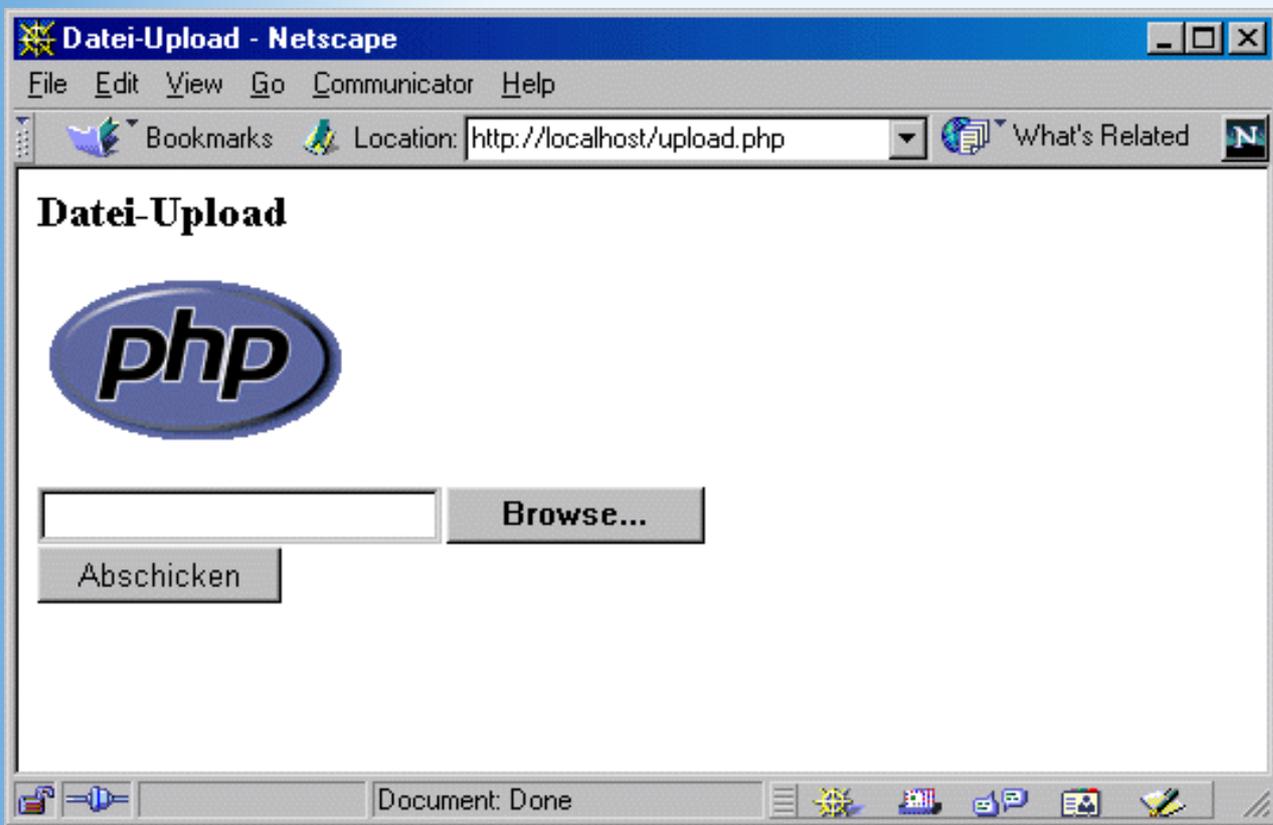
```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Datei-Upload</title>
</head>
<body bgcolor="white">
<h3>Datei-Upload</h3>
<p>
<?php
 if (is_uploaded_file($_HTTP_POST_FILES["datei"]["tmp_name"])) {
 $dateiname = $_HTTP_POST_FILES["datei"]["name"];
 move_uploaded_file($_HTTP_POST_FILES["datei"]["tmp_name"],
 "c:\\inetpub\\wwwroot\\$dateiname");
 print "";
 }
?>
</p>
<p>
<form method="post" enctype="multipart/form-data">
<input type="file" name="datei" />

<input type="submit" name="submit" value="Abschicken" />
</form>
</p>
</body>
</html>

```

Die Funktion `is_uploaded_file` überprüft, ob eine angegebene Datei auch wirklich via HTTP-File-Upload an den Server übertragen worden ist - ein weiteres Sicherheitsfeature.



**Abbildung 19.14: Die Grafik wurde vom Benutzer auf den Webserver übertragen**



*Achten Sie darauf, übertragene Grafiken in einem eigenen Verzeichnis abzulegen, damit Sie nicht eigene Dateien überschreiben!*

Wenn Sie mehrere Dateien auf einmal übertragen möchten, müssen Sie allen Datei-Upload-Formularelementen dasselbe name-Attribut geben, und - ähnlich wie bei den Auswahllisten - muss dieses auf [] enden. Sie können dann auf die einzelnen Attribute über den Index zugreifen: `$HTTP_POST_FILES["tmp_name"][0]`, `$HTTP_POST_FILES["tmp_name"][1]`, ...



*Weitere Informationen zum Datei-Upload unter PHP finden Sie im Online-Manual unter [www.php.net/manual/features.file-upload.php](http://www.php.net/manual/features.file-upload.php).*

## **Versende-Schaltflächen**

Auch bei diesem Formularelement gilt: Sag mir, wie Du heißt (name-Attribut), und ich sag Dir, wer Du bist (value-Attribut). Sie können das name-Attribut dazu verwenden, um herauszufinden, ob eine Seite direkt per URL aufgerufen worden ist, oder ob ein Formular an die aktuelle Seite

versandt worden ist (analog zum gestrigen Tag, ASP):

```
<?php
 if ($HTTP_POST_VARS["Versendebutton"] == "Abschicken") {
 // Formulardaten verarbeiten
 } else {
 // Formular anzeigen
 }
?>
```

## Praxisbeispiele

Nachdem Sie nun einen Überblick über alle Formularelemente sowie ihre Behandlung in PHP gewonnen haben, können (und sollten) Sie nun Ihr Wissen in der Praxis anwenden. Im Folgenden ein paar exemplarische Aufgabenstellungen und ihre Lösung - mit PHP.

### Anzeige aller übertragenen Daten

Um alle Formulardaten anzuzeigen, können Sie natürlich eine simple for-Schleife oder eine foreach-Schleife verwenden, aber Sie haben größere Möglichkeiten der Einflussnahme, wenn Sie auf die einzelnen Elemente direkt zugreifen.

#### Listing 19.21: daten\_anzeigen.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Formulare</title>
</head>
<body bgcolor="white">
<?php
 if ($HTTP_POST_VARS["submit"] == "Abschicken") :
?>
<h3>Formularwerte</h3>
<p>
<?php
 switch ($HTTP_POST_VARS["Geschlecht"]) {
 case "m":
 $anrede = "Herr"; break;
 case "f":
 $anrede = "Frau"; break;
 default:
 $anrede = "Herr/Frau";
 }
 print "Hallo $anrede ";
 print $HTTP_POST_VARS["Vorname"] . " ";
 print $HTTP_POST_VARS["Nachname"] . "!
"
```

?>

Sie setzen die folgenden Betriebssysteme ein:

```
<?php
 if (isset($_HTTP_POST_VARS["Windows"])) {
 print "Windows ";
 }
 if (isset($_HTTP_POST_VARS["Linux"])) {
 print "Linux ";
 }
}
```

?>

<br />Sch&ouml;n, dass Ihnen Tag

```
<?php print $_HTTP_POST_VARS["Tag"]; ?>
```

so gut gef&auml;llt.

</p>

```
<?php
 else :
```

?>

<h3>Formular (POST)</h3>

<p>

```
<form method="post">
```

```
<input type="text" name="Vorname" size="20" />Vorname

```

```
<input type="text" name="Nachname" size="20" />Nachname

```

```
<input type="radio" name="Geschlecht" value="m">männlich
```

```
<input type="radio" name="Geschlecht" value="f">weiblich

```

Ich nutze

```
<input type="checkbox" name="Windows" value="ja">Windows
```

```
<input type="checkbox" name="Linux" value="ja">Linux

```

```
<select name="Tag" size="1">
```

```
<?php
```

```
 for ($i=1; $i<=21; $i++) :
```

?>

```
<option value="<?=$i ?>"><?=$i ?></option>
```

```
<?php
```

```
 endfor;
```

?>

```
</select>Dieser Tag gefällt mir am Besten

```

```
<input type="submit" name="submit" value="Abschicken" />
```

```
</form>
```

</p>

```
<?php
 endif;
```

?>

```
</body>
```

```
</html>
```

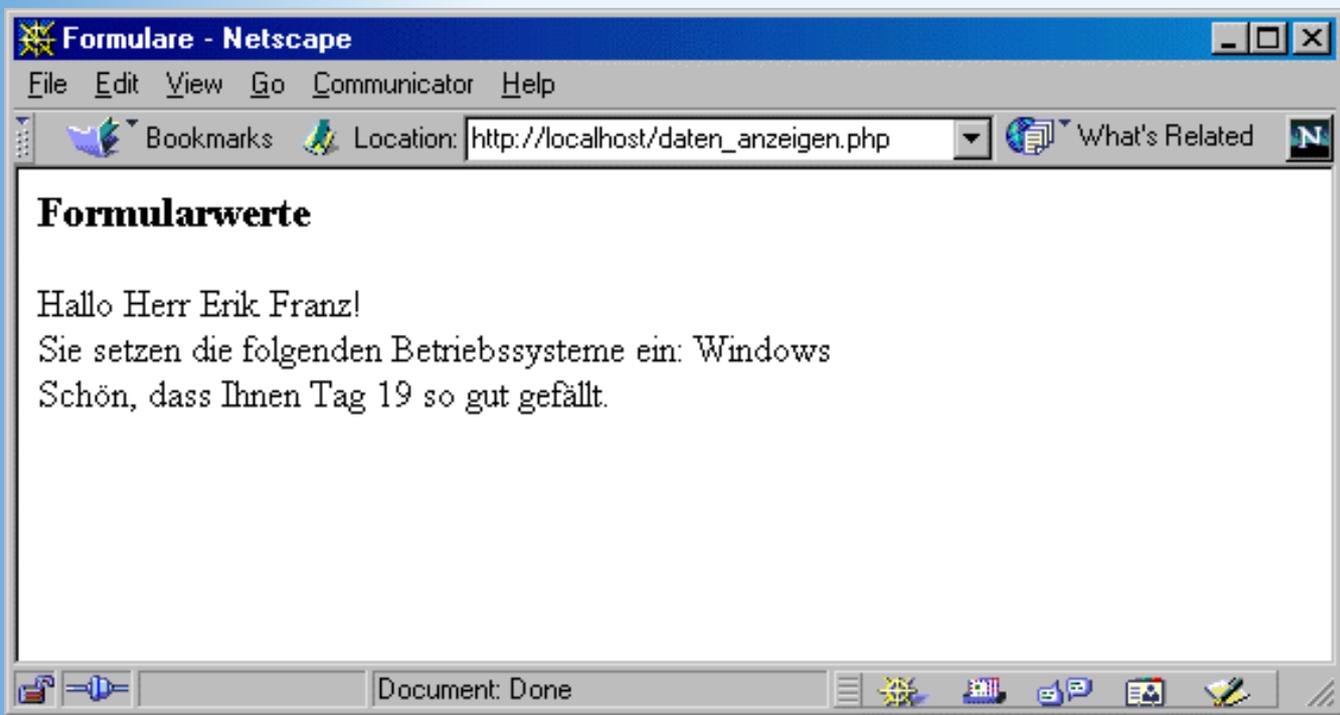


Abbildung 19.15: Die Formulare Daten werden ausgelesen und interpretiert

## Vollständigkeitsüberprüfung

Daten ausgeben ist eine Sache - aber da gibt es interessantere Anwendungsgebiete. Oft zitiert, oft implementiert und auch gestern schon einmal ein Thema war die Vollständigkeitsüberprüfung von Formularangaben. Nur vollständige Daten sind marketingrelevante Daten, und mit PHP haben Sie ein probates Mittel, zumindest einmal zu überprüfen, ob nicht etwas fehlt. Eine Logiküberprüfung, ob die eingegebenen Daten auch sinnvoll sind, ist ungleich komplexer, und in manchen Bereichen auch so gut wie unmöglich.

### Listing 19.22: vollstaendig.php

```
<?php
 $vollstaendig = false;
 if ($HTTP_POST_VARS["submit"] == "Abschicken") {
 $vollstaendig = true;
 if (!isset($HTTP_POST_VARS["Geschlecht"])) {
 $vollstaendig = false;
 }
 if (!isset($HTTP_POST_VARS["Vorname"])) {
 $vollstaendig = false;
 }
 if (!isset($HTTP_POST_VARS["Nachname"])) {
 $vollstaendig = false;
 }
 if (!isset($HTTP_POST_VARS["Windows"]) &&
 !isset($HTTP_POST_VARS["Linux"])) {
 $vollstaendig = false;
 }
 }
}
```

```

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Formulare</title>
</head>
<body bgcolor="white">
<h3>Formularversand</h3>
<p>
<?php
 if ($vollstaendig) :
?>
Danke für Ihre Angaben
<?php
 else :
?>
<form method="post">
<input type="text" name="Vorname" size="20" />Vorname

<input type="text" name="Nachname" size="20" />Nachname

<input type="radio" name="Geschlecht" value="m">männlich
<input type="radio" name="Geschlecht" value="f">weiblich

Ich nutze
<input type="checkbox" name="Windows" value="ja">Windows
<input type="checkbox" name="Linux" value="ja">Linux

<select name="Tag" size="1">
<?php
 for ($i=1; $i<=21; $i++) :
?>
<option value="<?=$i ?"><?=$i ?></option>
<?php
 endfor;
?>
</select>Dieser Tag gefällt mir am Besten

<input type="submit" name="submit" value="Abschicken" />
</form>
<?php
 endif;
?>
</p>
</body>
</html>

```

Anstelle der lapidaren Danksagung für das Ausfüllen des Formulars können Sie natürlich den Benutzer auch auf eine entsprechend gestaltete Dankesseite umleiten. Sie benötigen dazu die folgende Funktion:

```

<?php
 header("Location: Umleitungs-URL");

```

?>



*Beachten Sie, dass Sie - wie bei `Response.Redirect` in ASP - dieses Kommando vor der allerersten HTML-Ausgabe setzen müssen, sonst gibt es eine Fehlermeldung!*

Aus Platzgründen hier nur der Code vor `<!DOCTYPE`; der restliche Code bleibt im Vergleich zu obigem Listing unverändert.

### Listing 19.23: weiterleitung.php (Ausschnitt)

```
<?php
 $vollstaendig = false;
 if ($HTTP_POST_VARS["submit"] == "Abschicken") {
 $vollstaendig = true;
 if (!isset($HTTP_POST_VARS["Geschlecht"])) {
 $vollstaendig = false;
 }
 if (!isset($HTTP_POST_VARS["Vorname"])) {
 $vollstaendig = false;
 }
 if (!isset($HTTP_POST_VARS["Nachname"])) {
 $vollstaendig = false;
 }
 if (!isset($HTTP_POST_VARS["Windows"]) &&
 !isset($HTTP_POST_VARS["Linux"])) {
 $vollstaendig = false;
 }
 if ($vollstaendig) {
 header("Location: danke.php");
 }
 }
?>
```

## 19.5 Mit Dateien arbeiten

PHP bietet mannigfaltige Methoden des Dateizugriffs. Beispielsweise können Sie nicht nur auf lokale Dateien zugreifen, sondern auch auf Dateien, die Sie über HTTP oder FTP erreichen können. Wir wollen uns an dieser Stelle jedoch auf eine kleine Praxisanwendung beschränken: Das Schreiben von Formulardaten in eine Textdatei. Dies ist ein kostengünstiger Ersatz zu einer Datenbank, welche viele Hostler nur gegen Aufpreis anbieten. Sollten Sie am Thema Datenbanken interessiert sein - übermorgen erfahren Sie mehr!

Um in eine Datei zu schreiben, müssen Sie zunächst eine Verbindung zu dieser Datei öffnen. Sie

haben dazu zwei Möglichkeiten:

- `$datei = fopen("datei.txt", "w");` - öffnet die Datei, erstellt Sie wenn nötig, und überschreibt deren Inhalt.
- `$datei = fopen("datei.txt", "w+");` - öffnet die Datei, erstellt Sie wenn nötig, und hängt Daten ans Dateieende an.



*Wie immer bei Dateioperationen der wichtige Hinweis: Sie benötigen Schreibrechte in das Verzeichnis, in dem Sie Dateien anlegen wollen!*

Nach einer der obigen beiden Operationen enthält die Variable `$datei` einen numerischen Wert, das sogenannte *Datei-Handle*. Über diesen Wert können Sie nun in die Datei schreiben. Dazu dient die Funktion `fputs`, die zwei Parameter erwartet: Das Datei-Handle, sowie den zu schreibenden Text. Mit `fclose` wird das Datei-Handle wieder geschlossen.

Kommen wir nun zu obigem Formular zurück. Aufgabe ist es, die Werte, die ins Formular eingegeben werden, in eine Textdatei zu schreiben. Als Trennzeichen zwischen den einzelnen Werten dient der Tabulator (`»t«`), denn dann können Sie die Textdatei einfach in Excel, Access etc. importieren:

### Listing 19.24: dateizugriff.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Formulare</title>
</head>
<body bgcolor="white">
<?php
 if ($HTTP_POST_VARS["submit"] == "Abschicken") :
 $datei = fopen("form.txt", "a");
 fputs($datei, $HTTP_POST_VARS["Vorname"]."\t");
 fputs($datei, $HTTP_POST_VARS["Nachname"]."\t");
 fputs($datei, $HTTP_POST_VARS["Geschlecht"]."\t");
 fputs($datei, $HTTP_POST_VARS["Windows"]."\t");
 fputs($datei, $HTTP_POST_VARS["Linux"]."\t");
 fputs($datei, $HTTP_POST_VARS["Tag"]."\r\n");
 fclose($datei);
 ?>
<p>Vielen Dank für Ihre Angaben!</p>
<?php
 else :
```

```

?>
<h3>Formular (POST)</h3>
<p>
<form method="post">
<input type="text" name="Vorname" size="20" />Vorname

<input type="text" name="Nachname" size="20" />Nachname

<input type="radio" name="Geschlecht" value="m">männlich
<input type="radio" name="Geschlecht" value="f">weiblich

Ich nutze
<input type="checkbox" name="Windows" value="ja">Windows
<input type="checkbox" name="Linux" value="ja">Linux

<select name="Tag" size="1">
<?php
 for ($i=1; $i<=21; $i++) :
?>
<option value="<?=$i ?"><?=$i ?></option>
<?php
 endfor;
?>
</select>Dieser Tag gefällt mir am Besten

<input type="submit" name="submit" value="Abschicken" />
</form>
</p>
<?php
 endif;
?>
</body>
</html>

```

## 19.6 Cookies

Erinnern Sie sich noch an gestern? Die mühsame Unterscheidung zwischen dem Setzen und Lesen von Cookies, und Sie mussten sich immer überlegen, ob Sie jetzt auf das Request-Objekt oder das Response-Objekt zugreifen? Nun, bei PHP ist das nicht großartig anders, aber Sie finden es vermutlich intuitiver gelöst.

Wir gehen jetzt einmal davon aus, dass Sie gestern den Abschnitt über Cookies vollständig durchgearbeitet haben und somit auch das notwendige Grundwissen besitzen. Falls nicht, blättern Sie noch einmal zurück, und werfen Sie auch noch einen Blick auf die offizielle Cookie-Spezifikation von Netscape, die Sie unter [www.netscape.com/newsref/std/cookie\\_spec.html](http://www.netscape.com/newsref/std/cookie_spec.html) finden. Dann können Sie loslegen!

### Cookies setzen

Zum Setzen von Cookies dient die Funktion `setcookie`. Da Cookies als Teil des HTTP-Headers übertragen werden, gilt hier dieselbe Einschränkung wie auch bei der zuvor einmal eingesetzten `header`-Funktion: Sie muss jeweils vor der ersten HTML-Ausgabe aufgerufen werden!

Die Syntax der Funktion ist die folgende:

```
<?php
 setcookie(Name, Wert, Verfallsdatum, Pfad, Domain, Sicher)
?>
```

Von den Parametern sind in der Regel nur maximal die ersten drei interessant; dennoch führen wir hier den Sinn und Zweck aller Parameter auf:

- Der erste Parameter gibt den Namen des Cookies an. Dieser Parameter ist Pflicht.
- Der zweite Parameter gibt den Wert des Cookies an. Dieser Parameter ist optional.
- Der dritte Parameter gibt das Verfallsdatum des Cookies an. Cookies ohne Verfallsdatum werden automatisch gelöscht, sobald der Browser geschlossen wird. Cookies mit Verfallsdatum in der Vergangenheit werden sofort gelöscht. Dieser Parameter ist optional.
- Der vierte Parameter gibt an, unterhalb welchen Pfades der Cookie für den Webserver lesbar ist. Wenn Sie diesen Wert etwa auf »/markt« setzen, kann der Cookie von Seiten unterhalb */technik* nicht gelesen werden. Als Standardwert wird immer das aktuelle Verzeichnis genommen. Dieser Parameter ist optional.
- Der fünfte Parameter gibt an, innerhalb welcher Domain der Cookie sichtbar sein darf. Die Domain muss dabei mindestens zwei Punkte enthalten und kann nach links ergänzt werden. Ist der Wert beispielsweise »mut.de«, so können die Server *www.mut.de*, *www2.mut.de* und *mut.de* den Cookie lesen; bei »www.mut.de« eben nur *www.mut.de*. Dieser Parameter ist optional.
- Der sechste Parameter gibt an, ob der Cookie über eine gesicherte Verbindung (HTTPS) geschickt werden darf. Der Standardwert 0 bedeutet nein, 1 bedeutet ja. Dieser Parameter ist optional.

An einem Beispiel ausprobiert: Folgende Seite speichert die Lieblingsfarbe des Benutzers in einem (temporären) Cookie.

### Listing 19.25: cookie\_schreiben.php

```
<?php
 if (isset($_HTTP_POST_VARS["submit"])) {
 setcookie("Farbe", $_HTTP_POST_VARS["Farbe"]);
 }
?>

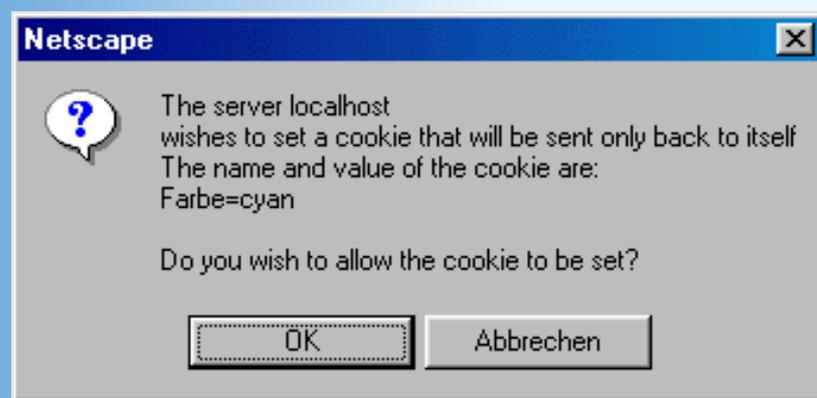
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Cookies</title>
</head>
<body bgcolor="white">
<h2>Cookies</h2>
<?php
 if (isset($_HTTP_POST_VARS["submit"])) :
?>
```

```

Sie haben <?php print $HTTP_POST_VARS["Farbe"]; ?>
als Ihre Lieblingsfarbe angegeben!
<?php
 else :
?>
<form method="post">
<input type="text" name="Farbe" size="20" /> Lieblingsfarbe

<input type="submit" name="submit" value="Abschicken">
</form>
<?php
 endif;
?>
</body>
</html>

```



**Abbildung 19.16: Der Server will einen Cookie an den Browser schicken**

Wenn der Cookie permanent gespeichert werden soll, müssen Sie ein Verfallsdatum angeben (im dritten Parameter). Am Einfachsten ist es, wenn Sie das aktuelle Datum nehmen und eine gewisse Zeitspanne hinzuaddieren:

```

<?php
 setcookie("Farbe", $HTTP_POST_VARS["Farbe"], time() + 60*60*24*30);
?>

```

Wozu das komische Produkt im dritten Parameter, und warum nicht direkt das Ergebnis, in diesem Falle 2592000? Nun, das dient der Übersichtlichkeit. Die Funktion `time` liefert die Epoche zurück, also die seit Anfang 1970 vergangenen Sekunden. Sie müssen zu diesem Wert die Sekunden addieren, die der Cookie ab dem momentanen Zeitpunkt noch gültig sein soll. Ein Tag hat 24 Stunden à 60 Minuten à 60 Sekunden, also entspricht  $60*60*24*30$  der Zahl Sekunden, die in 30 Tagen verstreichen. Das Cookie ist also 30 Tage lang gültig.

Um ein Cookie zu löschen, müssen Sie nur ein Datum in der Vergangenheit wählen, also beispielsweise im Verfallsdatum den gestrigen Tag wählen:

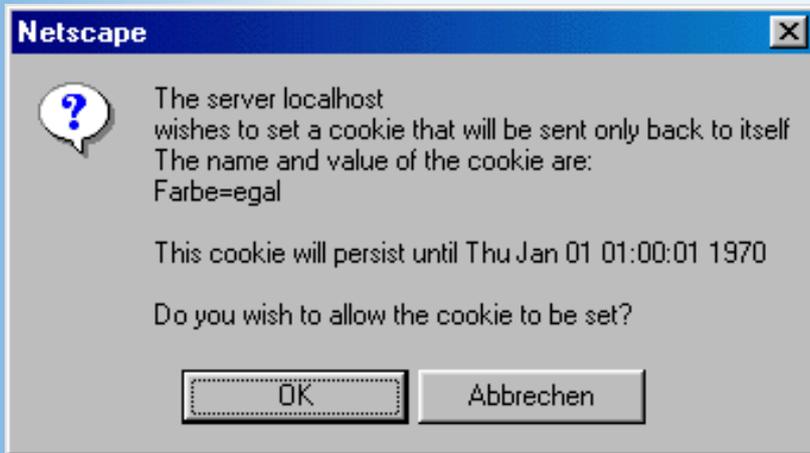
```

<?php
 setcookie("Farbe", "egal", time() - 60*60*24);
?>

```



Wenn Sie die Parameter für **Pfad**, **Domain** und **Sicher** beim Setzen eines Cookies angeben, müssen Sie diese exakt so auch beim Löschen des Cookies angeben!



**Abbildung 19.17: Der Cookie soll wieder gelöscht werden**

## Cookies lesen

Wir hatten es zuvor bereits einmal angedeutet - der Name des Cookies ist gleichzeitig der Variablenname, unter dem Sie auf den Wert des Cookies zugreifen können. Vorausgesetzt wird, dass **register\_globals** in der *php.ini* auf **On** gesetzt ist. Was aber immer funktioniert und was Sie aus den mehrfach angeführten Sicherheitsgründen unbedingt verwenden sollten, ist der Zugriff über das assoziative Array `$HTTP_COOKIE_VARS`. Der Name des Cookies dient hier als Schlüssel für den Wert.

```
<?php
 print "Lieblingsfarbe: " . $HTTP_COOKIE_VARS["Farbe"];
?>
```

## Praxisbeispiel

Auch dieses Mal wärmen wir ein Beispiel vom gestrigen Tage neu auf - denn so werden Unterschiede zwischen den verschiedenen serverseitigen Technologien deutlich erkennbar. Der Benutzer soll die Möglichkeit haben, seine Lieblingsfarbe in ein Formular einzugeben; sobald er das getan hat, wird diese Farbe als Hintergrundfarbe der Website eingesetzt. Das ist eine besonders einfache Form der Personalisierung, und legt damit den Grundstock für weitere, komplexere Personalisierungsaufgaben.

Beachten Sie auch hier die Hinweise von gestern: Die Farbe muss auf spitze öffnende und schließende Klammern überprüft bzw. diese aus dem String entfernt werden. Sodann kann der Cookie gesetzt werden.

Nachfolgend schon das Listing; da wir keine neuen Technologien verwenden und den Code schon gestern konzipiert und entwickelt hatten, bedarf es hier keiner weiteren Erklärungen.

### Listing 19.26: Lieblingsfarbe.php

```
<?php
 $farbe = $_HTTP_COOKIE_VARS["Farbe"];
 $farbe = str_replace("<", "", $farbe);
 $farbe = str_replace(">", "", $farbe);
 if ($farbe == "") {
 $farbe = "white";
 }
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Personalisierung</title>
</head>
<body bgcolor="<?=$farbe ?>">
<h2>Willkommen auf Ihrer personalisierten Seite!</h2>
<p>Ihre Lieblingsfarbe ist: <?=$farbe ?>.</p>
</body>
</html>
```



**Abbildung 19.18:** In der Schwarzweißabbildung kaum zu erkennen: Die Hintergrundfarbe

Damit wollen wir diesen Tag auch schon beenden. Es war relativ anstrengend, aber wie wir hoffen, auch sehr interessant und spannend. Weiterführende Literatur zum Thema PHP finden Sie in der Einleitung zum heutigen Tag. Bevor Sie sich aber weiter in die Materie stürzen, sollten Sie zunächst den Fragen- und Übungsteil erfolgreich hinter sich bringen!

## 19.7 Fragen und Antworten

**Frage:**

**Wie kann ich in PHP auf Umgebungsvariablen zugreifen?**

*Antwort:*

*Verwenden Sie die Funktion `getenv`. Beispielsweise erhalten Sie mit `getenv("HTTP_REFERER")` die URL der Seite, von der aus der Benutzer kam.*

**Frage:**

**Wie kann ich einen numerischen Wert in einen Integerwert umwandeln und umgekehrt?**

*Antwort:*

*PHP bietet hierzu Funktionen an, die Sie in der Dokumentation finden - aber auch hier funktioniert der kleine Trick von gestern: Integers werden zu Strings, wenn Sie die Integervariable mit »« konkatenieren. Strings werden zu Integer- Werten (oder auch zu Floats), wenn Sie sie mit 1 multiplizieren.*

**Frage:**

**Kann ich mit PHP ein wenig HTML-Code ausgeben, und dann nach ein paar Sekunden den Benutzer auf eine andere Seite weiterleiten?**

*Antwort:*

*Nein, das geht nur mit clientseitigen Mitteln. In den vorhergehenden beiden Wochen haben Sie dazu Mittel und Wege kennen gelernt, dies sowohl mit HTML als auch mit JavaScript zu realisieren.*

**Frage:**

**Kann ich PHP 3 und PHP 4 parallel installieren?**

*Antwort:*

*Das ist prinzipiell möglich, da die Ini-Datei von PHP 3 `php3.ini` heißt. Doch obwohl PHP 4 wohl etwas zu früh veröffentlicht worden ist (Zur Drucklegung des Buches ist bereits Unterversion 4.0.4 veröffentlicht, die schon deutlich stabiler ist als 4.0.0), gibt es mittlerweile kaum einen Grund mehr, noch PHP 3 zu verwenden.*

**Frage:**

**Ich will per HTTP/FTP auf eine Datei zugreifen, aber mit dem Parameter »a« oder »w« für `fopen` funktioniert es nicht. Woran liegt das?**

*Antwort:*

*Sie benötigen hier expliziten Lesezugriff. Dazu dient der Parameter »r«. Nähere Informationen finden Sie im Online-Manual unter der Funktion `fopen`.*

**Frage:**

**Warum kann ich nicht unmittelbar, nachdem ich ein Cookie gesetzt habe, darauf zugreifen?**

*Antwort:*

*Der Cookie muss erst an den Client geschickt und dort gespeichert werden. Sie können also nicht serverseitig darauf zugreifen, denn der Client hat es noch nicht an den Server schicken können. Merken Sie sich also den Wert im Cookie, und beim Laden der nächsten Seite wird der Cookie zu Verfügung stehen.*

**Frage:**

**Wofür steht eigentlich PHP?**

*Antwort:*

*Hierzu gibt es viele falsche Antworten - und eine richtige. PHP steht für: »PHP: Hypertext Preprocessor«.*

## 19.8 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

4. Was ist Zend, und was hat das mit Rasmus Lerdorf zu tun?
5. Ist das Gleichheitszeichen im PHP ein Vergleichs- oder ein Zuweisungsoperator?
6. Was ist der Hauptunterschied zwischen if und switch?
7. Mit welchen Schleifen können Sie besonders bequem auf alle Arrayelemente zugreifen? Welche Parameter sind hier wichtig?
8. Wozu dient \$PHP\_SELF?
9. Wie greifen Sie auf alle ausgewählten Elemente einer Auswahlliste zu, die mit <select multiple> erstellt worden ist? Was müssen Sie im HTML-Code beachten?
10. Wozu ist der zweite Parameter von fopen gut?
11. Welche Parameter von setcookie sind nicht optional?

### Übungen

1. Begrüßen Sie auf einer PHP-Seite Ihre Besucher je nach Tageszeit (»Guten Morgen« etc.)!
2. Erweitern Sie die Vollständigkeitsüberprüfung des Formulars aus diesem Kapitel! Momentan ist es so, dass bei nicht vollständiger Ausfüllung das Formular erneut angezeigt wird, aber komplett leer. Sorgen Sie dafür, dass bereits eingegebene Werte stehen bleiben.
3. Fassen Sie die Listings zum Thema Personalisierung zusammen: Auf einer PHP-Seite erscheint entweder die angegebene Hintergrundfarbe, oder der Benutzer wird zu ihrer Eingabe aufgefordert.



## Tag 20

# Datenbankanbindung

Heute werden wir uns mit dem Thema Datenbanken beschäftigen. Die Arbeit mit Datenbanken ist für professionelle Websites ein Quasi-Muss, und das Ganze ist eigentlich gar nicht so schwer, wenn man nur einmal weiß, wie. Am heutigen Tage weihen wir Sie in die Geheimnisse der Datenbanksteuerung ein.

Die Themen heute:

- Was ist eine Datenbank?
- Konzeption einer relationalen Datenbank
- Einführung in SQL
- Ansteuerung via ODBC und ASP
- Ansteuerung via ODBC und PHP
- PHP und MySQL
- Alternativen

## 20.1 Was ist eine Datenbank?

Der Begriff »Datenbank« ist selbstbeschreibend. Unter einer Datenbank versteht man einen Speicher für Daten. Und zwar für eine ganze Menge von Daten. Diese Daten sind in einer bestimmten Struktur angeordnet. Im Folgenden erfahren Sie eine ganze Reihe von Dingen über Datenbanken im Allgemeinen. Zwar hat jede Datenbank ihre Eigenheiten, Spezialitäten sowie individuellen Vor- und Nachteile, aber das meiste, was Sie nun lesen werden, ist von der letztendlich verwendeten Datenbank völlig unabhängig. Aber keine Angst vor der grauen Theorie, in der zweiten Hälfte des heutigen Tages werden wir in die Praxis übergehen und unterschiedliche Datenbanken mit ASP und PHP ansteuern.



*Warum wir kein Perl verwenden? Nun, bei ASP und PHP ist die Datenbankunterstützung mehr oder weniger mit eingebaut, bei Perl benötigen Sie externe Module, und deren Ansteuerung ist - je nach verwendeter Datenbank - unterschiedlich. Und da Perl eh bei der Webprogrammierung auf dem absterbenden Ast ist (nicht nur unserer Meinung nach), sollten auch Sie für fortschrittliche Programmierung Ihr Hauptaugenmerk auf ASP und PHP richten.*

Die Daten innerhalb einer Datenbank sind in Tabellen angeordnet. Die Tabelle wiederum besteht aus mehreren Zeilen und Spalten. Wenn Sie bereits einmal mit einer Tabellenkalkulation gearbeitet haben (beispielsweise Excel oder Quattro Pro), kennen Sie das Prinzip möglicherweise. Jede Tabellenspalte hat ein bestimmtes Format, enthält beispielsweise einen String, eine Zahl oder ein Datum.

## Datenmodellierung

Vor jeder Programmierung steht die Modellierung der Datenbank. Im Folgenden werden wir das einmal exemplarisch vorführen, und dabei auch unterschiedliche Ansätze präsentieren. Wir werden jedoch ganz klar sagen, wo unsere persönlichen Präferenzen liegen.

Aber nun zu einem konkreten Beispiel, und ganz im Zeichen des E-Commerce nehmen wir als Beispiel einen Online-Shop. In diesem Online-Shop werden eine Reihe von Waren verkauft, und die Bestellungen sollen in einer Datenbank abgespeichert werden. Zunächst überlegen wir uns, welche Daten überhaupt in der Datenbank gehalten werden müssen:

- Name des Bestellers
- Adresse des Bestellers
- Bestellte Artikel
- Bestelldatum
- Auftragsnummer

Ein erster Ansatz für den Datenbankaufbau könnte ungefähr folgendermaßen aussehen:

Name	Datentyp	Beschreibung
auftragsnummer	String oder Zahl	Auftragsnummer
vorname	String	Vorname des Bestellers
nachname	String	Nachname des Bestellers
strasse	String	Straßenanschrift des Bestellers
plz	String	Postleitzahl des Bestellers
ort	String	Wohnort des Bestellers
artikel	String	Bestellte Artikel, nebst Anzahl
datum	Datum	Datum der Bestellung

**Tabelle 20.1: Ein erster Ansatz für unsere E-Commerce-Datenbank**



*Sie sehen hier gleich eine häufig verwendete Konvention für Spaltennamen: Nur Kleinbuchstaben!*

Dieser Ansatz funktioniert tatsächlich - in der Praxis werden Sie ihn jedoch so gut wie nie antreffen. Dieses Datenbankmodell weist nämlich *Redundanz* auf - das heißt, Informationen werden zu oft gespeichert. Stellen Sie sich vor, ein Kunde bestellt mehrmals im Online-Shop. Dann befinden sich in der Tabelle mehrere Einträge - Zeilen -, die eine Bestellung dieses Kunden beschreiben. In jeder dieser Zeilen steht der Name des Kunden sowie seine Anschrift - und bei jeder Bestellung somit das Gleiche. Und dies nennt man Redundanz.

## Relationales Datenbankmodell

Aus diesem Grund verwendet man bei der Datenbankmodellierung heutzutage zumeist ein *relationales* Datenbankmodell. Anstelle von einer einzigen, großen Tabelle, teilt man die Daten auf mehrere Tabellen auf, die in einer bestimmten Art und Weise miteinander verknüpft sind. In unserem Beispiel lohnt es sich also, eine Tabelle für die Kunden einzurichten. Diese Tabelle habe den Namen *kunden*:

Name	Datentyp	Beschreibung
kundennummer	String oder Zahl	Kundennummer
vorname	String	Vorname des Kunden
nachname	String	Nachname des Kunden
strasse	String	Straßenanschrift des Kunden
plz	String	Postleitzahl des Kunden
ort	String	Wohnort des Kunden

**Tabelle 20.2: Die Tabelle *kunden***

Wie Sie sehen, hat jeder Kunde eine Kundennummer, die natürlich eindeutig sein muss. In anderen Worten, jeder Kunde hat eine andere Kundennummer. In der Spalte *kundennummer* steht also ein eindeutiger Bezeichner für den Eintrag in der Tabelle. Dies nennt man auch einen *Primärschlüssel*.

Die zuvor entwickelte Tabelle kann nun wie folgt vereinfacht werden:

Name	Datentyp	Beschreibung
auftragsnummer	String oder Zahl	Auftragsnummer
kundennummer	String oder Zahl	Kundennummer des Bestellers
artikel	String	Bestellte Artikel, nebst Anzahl
datum	Datum	Datum der Bestellung

**Tabelle 20.3: Die Tabelle *bestellungen***

In der Spalte *kundennummer* steht die Kundennummer des Bestellers. Diese Kundennummer ist der Primärschlüssel in der Tabelle *kunden*, das heißt also, über die Angabe dieser Nummer kann auf den gesamte Adressdatensatz des Kunden zugegriffen werden. Man sagt in diesem Fall auch, die Spalte *kundennummer* der Tabelle *bestellungen* enthält einen *Fremdschlüssel*. Zusammenfassend: Ein Fremdschlüssel ist ein Primärschlüssel einer anderen Tabelle.

Noch eine weitere neue Begrifflichkeit. Es ist nun so, dass ein Kunde mehrere Bestellungen tätigen kann, jede Bestellung aber genau einem Kunden zugeordnet werden kann. Zwischen den Tabellen *kunden* und *bestellungen* gibt es damit eine sogenannte *1:n-Beziehung*.

Ein weiteres Problem am bisherigen Datenmodell liegt in der Abspeicherung der einzelnen, bestellten Artikel. Bis jetzt ist das durch ein einziges Feld gelöst, einem String. Ein möglicher Wert ist etwa »ein Buch mit der ISBN 3-8272-5764-6 zum Preis von 49,95 DM, sowie zwei Exemplare des Buchs mit der ISBN 90-430-0128-7«. Das ist natürlich im Sinne einer effizienten Warenwirtschaft völlig untragbar. Außerdem soll ja das Online- Bestellsystem auch aus einer Datenbank gefüttert werden. Es ist also eine

gute Idee, die Artikel, die bestellt werden können, auch in der Datenbank zu speichern. Die Idee liegt nahe, dazu eine eigene Tabelle anzulegen. Diese könnte folgendermaßen aussehen:

Name	Datentyp	Beschreibung
bestellnummer	String oder Zahl	Bestellnummer des Artikels
beschreibung	String	textuelle Beschreibung des Artikels
preis	Zahl	Preis des Artikels (in _ oder DM)

**Tabelle 20.4: Die Tabelle *artikel***

Die Frage ist nun, wie das mit der Tabelle *bestellungen* in Einklang zu bringen ist. Zwar könnte die Spalte *bestellnummer* (welche der Primärschlüssel der Tabelle *artikel* ist) als Fremdschlüssel eingebaut werden, aber Sie müssen immer noch festhalten, welcher Artikel wie oft bestellt worden ist. Manche Leute, die gerne Excel einsetzen (oder eine andere Tabellenkalkulation), legen nun beliebig viele Spalten an, und ermöglichen damit beliebig viele zu bestellende Artikel; in einer Datenbank ist das aber nicht mehr so ohne Weiteres möglich, da die Anzahl der Spalten im Vornherein feststehen muss. Zwischen den Tabellen *artikel* und *bestellungen* gibt es eine sogenannte *n:m-Beziehung*, denn ein Artikel kann Teil von mehreren Bestellungen sein, und eine Bestellung kann mehrere Artikel enthalten. In so einem Fall führt man eine weitere Tabelle ein, welche die Verbindung zwischen den Artikeln und den Bestellungen herstellt. Steht in einer Zeile dieser Tabelle die Bestellnummer X und der Artikel Y, dann bedeutet dies, das in der Bestellung X der Artikel Y bestellt worden ist. Um festzustellen, welche Artikel alle in Bestellung X bestellt worden sind, müssen Sie in der neuen Tabelle alle Zeilen heraussuchen, die die Bestellnummer X enthalten.

Einen Punkt haben wir noch vergessen: Es muss noch festgehalten werden, wie oft der jeweilige Artikel bestellt worden ist. Damit ist die Tabelle auch schon fertig. Als Name nimmt man oft die Namen der beiden Tabellen, die in einer *n:m-Beziehung* zueinander stehen, und trennt diese mit einem Unterstrich:

Name	Datentyp	Beschreibung
artikel_bestellungen_id	Zahl	Primärschlüssel
bestellnummer	String oder Zahl	Bestellnummer des bestellten Artikels
auftragsnummer	String oder Zahl	Auftragsnummer
anzahl	Zahl	Anzahl, wie oft der Artikel in der Bestellung enthalten ist

**Tabelle 20.5: Die Tabelle *artikel\_bestellungen***



*Sie werden bemerkt haben, dass diese Tabelle auch einen Primärschlüssel enthält, obwohl dieser nicht als Fremdschlüssel in einer anderen Tabelle verwendet wird. Es ist jedoch generell eine gute Praxis, jeder Tabelle einen Primärschlüssel zuzuweisen, und sei dies nur eine fortlaufende Nummer des Eintrags.*

Somit ist unser Datenmodell fertig. Die Tabelle *bestellungen* muss nur noch wie folgt angepasst werden - die bestellten Artikel stehen ja jetzt in einer anderen Tabelle:

Name	Datentyp	Beschreibung
auftragsnummer	String oder Zahl	Auftragsnummer
kundennummer	String oder Zahl	Kundennummer des Bestellers
datum	Datum	Datum der Bestellung

**Tabelle 20.6: Die Tabelle *bestellungen***

## Beispieldaten

Somit haben Sie in Tabelle 20.2, Tabelle 20.4, Tabelle 20.5 und Tabelle 20.6 ein vollständiges, und auch relativ funktionsfähiges Datenmodell erstellt, das wir auch im Folgenden weiterverwenden werden. Damit Sie sich das Ganze noch einmal direkt am Beispiel verdeutlichen können, hier einige Beispieldaten für die Datenbank. Beginnen wir mit der Tabelle *kunden*:

kundennummer	vorname	nachname	strasse	plz	ort
123456-abc	Erik	Franz	Martin-Kollar-Str. 10-12	81829	München
08/15	Bill	Gates	One Microsoft Way	12345	Redmond
K-2.4.0	Linus	Torvalds	Freedom Circle	67890	Santa Clara

**Tabelle 20.7: Beispieldaten für die Tabelle *kunden***

Die Tabelle *artikel*, welche die zu bestellenden Waren nebst Preis enthält (noch in DM), kann folgendermaßen gefüllt sein:

bestellnummer	beschreibung	preis
3-8272-6003-5	Louis/Wenz: Dynamic Web-Publishing in 21 Tagen	89,95
3-8272-5762-X	Louis/Pott: Frontpage 2000 Kompendium	89,95
3-8272-5764-6	Wenz/Hauser: Jetzt lerne ich Dynamic Web-Publishing	49,95

**Tabelle 20.8: Beispieldaten für die Tabelle *artikel***

Als nächstes wird die Tabelle *bestellungen* gefüllt. Dazu benötigen Sie die Kundennummern aus der Tabelle *kunden*.

auftragsnummer	kundennummer	datum
1	08/15	01.01.2001
2	K-2.4.0	13.03.2001

3	08/15	19.04.2001
---	-------	------------

**Tabelle 20.9: Beispieldaten für die Tabelle *bestellungen***

Damit sind die Bestellungen aber noch nicht vollständig erfasst - die Tabelle *artikel\_bestellungen* muss noch mit den Bestelldaten gefüllt werden:

artikel_bestellungen_id	bestellnummer	auftragsnummer	anzahl
1	3-8272-6003-5	1	1
2	3-8272-5764-6	1	1
3	3-8272-5762-X	2	3
4	3-8272-5762-X	3	5

**Tabelle 20.10: Beispieldaten für die Tabelle *artikel\_bestellungen***

Sie können nun beispielsweise erkennen, dass Linus Torvalds am 13. März 2001 drei Exemplare des »Frontpage 2000 Kompendiums« bestellt hat. Und diese Daten stehen jetzt nicht mehr in einer Tabelle, sondern sind in vier Tabellen verteilt. Der Lohn sind übersichtlichere, kleinere Tabellen, sowie weniger Redundanz. Aber gehen wir nun direkt in die Praxis über - wie bringen Sie Ihrer Datenbank bei, solche Tabellen anzulegen, und auch noch mit Daten zu füllen oder Informationen daraus auszulesen?

## 20.2 SQL

In den siebziger Jahren hat IBM, damals ein Pionier in der Datenbankentwicklung, eine standardisierte Abfragesprache entwickelt, um damit die verschiedensten Datenbanken anzusprechen, und zwar unabhängig vom Datenbanktyp oder -hersteller. Die erste Version dieser Sprache hieß SEQUEL, in der Folge hat man sich aber auf das kürzere SQL geeinigt - Structured Query Language. Die alte Form ist zumindest noch im Sprachgebrauch vorhanden, denn viele Leute sprechen »Microsoft SQL Server« aus wie »Microsoft SEQUEL Server«. Das spricht natürlich nicht unbedingt für die Englisch- Fähigkeiten von Microsoft-Anhängern; das Open-Source-Pendant MySQL wird wie erwartet ausgesprochen.

### Tabellen erstellen

Nachdem Sie Ihre Datenbankstruktur erstellt haben, müssen Sie zunächst die einzelnen Tabellen erzeugen. Dazu müssen Sie sich die Datentypen der einzelnen Spalten in den Tabellen überlegen. Nun bietet jeder Datenbankhersteller eigene Datentypen an, so dass Sie an den entsprechenden Stellen nachlesen müssen, was Ihre Datenbank unterstützt und was nicht.



*Ganz allgemein soll natürlich nicht verschwiegen werden, dass die meisten Datenbanken eine grafische Oberfläche bieten, mit der Sie die einzelnen Felder auch direkt erstellen können.*

In Tabelle 20.11 sehen Sie die häufigsten Datentypen, die von praktisch jeder Datenbank unterstützt werden:

Datentyp	Beschreibung
CHAR(n)	String mit genau $n$ Zeichen
DATETIME	Datumswert
DOUBLE	64 Bit langer Fließkommawert
INTEGER	Numerischer Wert (Integer)
VARCHAR(n)	String mit höchstens $n$ Zeichen

**Tabelle 20.11: Die häufigsten Datentypen**



*Der Unterschied zwischen CHAR und VARCHAR ist, dass bei CHAR immer die maximale Anzahl von Zeichen in der Datenbank reserviert wird; bei VARCHAR wird der Platz in Abhängigkeit vom aktuellen Wert in der Spalte belegt, womit dieser Datentyp bei variabel langen Einträgen effizienter ist.*

Um nun eine Tabelle zu erstellen, benötigen Sie das Kommando CREATE TABLE. Dazu geben Sie den Namen der Tabelle an, die Namen der Spalten sowie die entsprechenden Datentypen. Die Tabelle *artikel* könnte also wie folgt erstellt werden:

#### **Listing 20.1: create\_artikel.sql**

```
CREATE TABLE artikel(
 bestellnummer CHAR(13) PRIMARY KEY,
 beschreibung VARCHAR(255),
 preis DOUBLE
)
```

Durch die Angabe von PRIMARY KEY wird angegeben, dass die entsprechende Spalte Primärschlüssel ist.

Die Tabelle *kunden* wird ähnlich erzeugt:

#### **Listing 20.2: create\_kunden.sql**

```
CREATE TABLE kunden(
 kundenummer VARCHAR(20) PRIMARY KEY,
 vorname VARCHAR(20),
 nachname VARCHAR(20),
 strasse VARCHAR(30),
 plz CHAR(5),
 ort VARCHAR(20)
```

)



*In obiger Anweisung finden Sie einige Einschränkungen; beispielsweise sind für die Postleitzahl exakt fünf Zeichen reserviert. Das mag innerhalb von Deutschland noch sinnvoll sein, wenn Sie auch Kunden aus dem Ausland haben sollten Sie etwa VARCHAR(10) verwenden.*

Bei der Tabelle *bestellungen* gibt es zwei Neuerungen. Zunächst hat jeder Eintrag in der Tabelle eine numerische ID, welche als laufender Zähler realisiert ist. Praktisch alle Datenbanken unterstützen das sogenannte Autowert-Konzept und vergeben diese IDs auf Wunsch automatisch.

Zweite Besonderheit ist die Spalte *kundennummer*, welche Fremdschlüssel aus der Tabelle *kunden* ist. Dies muss durch die Anweisung CONSTRAINT Name FOREIGN KEY(kundennummer) REFERENCES kunden angezeigt werden.

### Listing 20.3: create\_bestellungen.sql

```
CREATE TABLE bestellungen(
 auftragsnummer INTEGER AUTO_INCREMENT PRIMARY KEY,
 kundennummer VARCHAR(20),
 datum DATETIME,
 CONSTRAINT bestellungen_fk1
 FOREIGN KEY(kundennummer) REFERENCES kunden
)
```

Durch INTEGER AUTO\_INCREMENT wird ein Integer-Autowert gekennzeichnet. Der erste Eintrag in der Tabelle erhält die ID 1, der nächste die ID 2, und so weiter.

Fehlt nur noch die Tabelle *artikel\_bestellungen*:

### Listing 20.4: create\_artikel\_bestellungen.sql

```
CREATE TABLE artikel_bestellungen(
 artikel_bestellungen_id INTEGER AUTO_INCREMENT PRIMARY KEY,
 bestellnummer CHAR(13),
 auftragsnummer INTEGER,
 anzahl INTEGER,
 CONSTRAINT artikel_bestellungen_fk1
 FOREIGN KEY(bestellnummer) REFERENCES artikel,
 CONSTRAINT artikel_bestellungen_fk2
 FOREIGN KEY(auftragsnummer) REFERENCES bestellungen
)
```



*Bevor wir uns nun der eigentlich interessanten und vor allem praxisrelevanteren Seite von SQL zuwenden, hier noch einmal unser Hinweis: Nicht jede Datenbank unterstützt den kompletten SQL-Satz, und in den wenigsten Datenbanken müssen Sie die Tabellen von Hand erstellen. Ihnen steht dazu eine grafische Benutzeroberfläche zur Verfügung, mit der Vieles einfacher geht.*

## In die Datenbank schreiben

Um eine Datenbank überhaupt erst mit Daten zu füllen, benötigen Sie die INSERT- Anweisung. Diese hat die folgende Syntax:

```
INSERT INTO Tabelle (
 Spalte1,
 Spalte2,
 ...
 SpalteX
) VALUES (
 Wert1,
 Wert2,
 ...
 WertX
)
```



*Die Zeilensprünge haben einen rein kosmetischen Effekt und dienen dazu, dass die SQL-Anweisung übersichtlicher wird. Sie können auch alles in eine Zeile schreiben.*

Strings müssen hierbei in einfache Anführungszeichen gefasst werden, numerische Werte benötigen keine Apostrophen (aber es funktioniert auch mit Apostrophen, wenngleich das nicht ganz sauber ist).

Um das Ganze einmal an einem realen Beispiel durchzuführen, füllen wir unsere Beispielstabellen mit den exemplarischen Werten von vorher. Beginnen wir mit der Tabelle *kunden*:

### Listing 20.5: insert\_kunden.sql

```
INSERT INTO kunden (
 kundenummer, vorname, nachname, strasse, plz, ort
) VALUES (
 '123456-abc', 'Erik', 'Franz', 'Martin-Kollar-Str. 10-12', '81829',
 'München'
)
INSERT INTO kunden (
 kundenummer, vorname, nachname, strasse, plz, ort
) VALUES (
 '08/15', 'Bill', 'Gates', 'One Microsoft Way', '12345', 'Redmond'
)
INSERT INTO kunden (
 kundenummer, vorname, nachname, strasse, plz, ort
) VALUES (
 '08/15', 'Bill', 'Gates', 'One Microsoft Way', '12345', 'Redmond'
)
```

```

 kundennummer, vorname, nachname, strasse, plz, ort
) VALUES (
 'K-2.4.0', 'Linus', 'Torvalds', 'Freedom Circle', '67890',
 'Santa Clara'
)

```

Als nächstes möchten wir die Tabelle *artikel* füllen. Besonderheit hier: Praktisch alle Datenbanken verwenden - auch aufgrund ihrer amerikanischen Herkunft - Dezimalpunkte anstelle von Dezimalkommata.

### Listing 20.6: insert\_artikel.sql

```

INSERT INTO artikel (
 bestellnummer, beschreibung, preis
) VALUES (
 '3-8272-6003-5', 'Louis/Wenz: Dynamic Web-Publishing in 21 Tagen',
 89.95
)
INSERT INTO artikel (
 bestellnummer, beschreibung, preis
) VALUES (
 '3-8272-5762-X', 'Louis/Pott: Frontpage 2000 Kompendium', 89.95
)
INSERT INTO artikel (
 bestellnummer, beschreibung, preis
) VALUES (
 '3-8272-5764-6',
 'Wenz/Hauser: Jetzt lerne ich Dynamic Web-Publishing', 49.95
)

```

Beim Schreiben in die Tabelle *bestellungen* gilt es zwei Dinge zu beachten: Zum einen müssen Sie der Spalte *auftragsnummer* keinen Wert zuweisen, denn dies macht die Datenbank automatisch (da Datentyp *AUTO\_INCREMENT*). Bei *kundennummer* müssen Sie unbedingt darauf achten, als Wert einen Primärschlüssel aus der Tabelle *kunden* zu verwenden, denn sonst ist die Fremdschlüsselbedingung verletzt und die Datenbank gibt eine Fehlermeldung aus.

Sie sehen hier, wozu ein Fremdschlüssel gut sein kann: Die Gefahr, dass fehlerhafte Daten in die Datenbank geschrieben werden können, wird verringert; die sogenannte *Datenintegrität* bleibt somit erhalten.

### Listing 20.7: insert\_bestellungen.sql

```

INSERT INTO bestellungen (
 kundennummer, datum
) VALUES (
 '08/15', '01.01.2001'
)
INSERT INTO bestellungen (
 kundennummer, datum
) VALUES (
 'K-2.4.0', '13.03.2001'
)
INSERT INTO bestellungen (

```

```
kundennummer, datum
) VALUES (
 '08/15', '19.04.2001'
)
```



*Datumswerte sind so eine Sache, insbesondere dann, wenn Sie die Datenbank von einem Skript aus ansteuern. Je nach Version und Hersteller der Datenbank sowie Ländereinstellungen des jeweiligen Betriebssystems muss das Datum in einem anderen Format übergeben werden. In Deutschland üblich ist tt.mm.jjjj, aber in Amerika steht der Monat vor dem Tag, also mm.tt.jjjj oder auch jjjj-mm-tt. Sie müssen also ein wenig herumexperimentieren.*

Als Letztes muss noch die Tabelle *artikel\_bestellungen* gefüllt werden:

### **Listing 20.8: insert\_artikel\_bestellungen.sql**

```
INSERT INTO artikel_bestellungen (
 bestellnummer, auftragsnummer, anzahl
) VALUES (
 '3-8272-6003-5', 1, 1
)
INSERT INTO artikel_bestellungen (
 bestellnummer, auftragsnummer, anzahl
) VALUES (
 '3-8272-5764-6', 1, 1
)
INSERT INTO artikel_bestellungen (
 bestellnummer, auftragsnummer, anzahl
) VALUES (
 '3-8272-6003-5', 1, 1
)
INSERT INTO artikel_bestellungen (
 bestellnummer, auftragsnummer, anzahl
) VALUES (
 '3-8272-5762-X', 2, 3
)
INSERT INTO artikel_bestellungen (
 bestellnummer, auftragsnummer, anzahl
) VALUES (
 '3-8272-6003-5', 3, 5
)
```

Da in der Tabelle *artikel\_bestellungen* als Fremdschlüssel die Primärschlüssel aus der Tabelle *bestellungen* verwendet werden, müssen Sie beim Schreiben einer neuen Bestellung in die Datenbank zunächst die Tabelle *bestellungen* füllen, dann *artikel\_bestellungen*.

Sie benötigen also den Autowert, der einer neuen Bestellung zugewiesen wird. Manche Datenbanken

bieten hierzu eine eigene Abfrage an, die diesen Autowert zurückliefert. In Abschnitt 20.2.4 zeigen wir Ihnen eine Alternative.

## Die Datenbank aktualisieren

Angenommen, Bill Gates will nicht mehr »Bill« genannt werden, sondern »William H.«. Kein Problem, sagt der Datenbankadministrator, ändern wir das einfach in der Datenbank. Das dazugehörige SQL-Kommando lautet UPDATE, und die Syntax ist folgendermaßen:

```
UPDATE Tabelle SET
 Spalte1 = Wert1,
 Spalte2 = Wert2,
 ...
 SpalteX = WertX
WHERE Bedingung
```

Beginnen wir mit einem einfachen Beispiel:

```
UPDATE kunden SET
 vorname = 'William H.'
```

Diese Anweisung würde das Feld *vorname* auf den Wert »William H.« setzen - und zwar bei allen Tabelleneinträgen. Das ist natürlich unerwünscht, denn es soll nur der Eintrag von Bill Gates geändert werden. Hierzu dient die WHERE-Bedingung:

```
UPDATE kunden SET
 vorname = 'William H.'
WHERE nachname = 'Gates'
```

Es lassen sich mehrere Bedingungen mit den booleschen Operatoren AND, OR und NOT miteinander verknüpfen, wie Sie das bereits von VBScript gewohnt sind:

```
UPDATE kunden SET
 vorname = 'William H.'
WHERE nachname = 'Gates' AND ort = 'Redmond'
```

Bei den Textvergleichen können Sie auch Jokerzeichen verwenden; anstelle des Gleichheitszeichens kommt dann der Operator LIKE zum Einsatz. SQL kennt die beiden folgenden Jokerzeichen:

- `_` - steht für ein beliebiges Zeichen
- `%` - steht für beliebig viele Zeichen



*Innerhalb von Access gibt es andere Jokerzeichen: ? steht für ein beliebiges Zeichen, \* für beliebig viele Zeichen. Wenn Sie Access über ODBC ansteuern (dazu weiter unten mehr), müssen Sie jedoch wieder die »originalen« SQL-Jokerzeichen verwenden.*

Die beiden folgenden Abfragen haben denselben Effekt, wenn Sie auf den Datenbestand aus Tabelle

20.7 angewandt werden:

```
UPDATE kunden SET
 vorname = 'William H.'
WHERE nachname LIKE 'G_t_s'
UPDATE kunden SET
 vorname = 'William H.'
WHERE nachname LIKE 'G%s'
```



*Sie sehen an dieser Stelle übrigens einen weiteren Vorteil eines relationalen Datenbankmodells. Hätten wir nur eine Tabelle, so müssten wir für jede Bestellung von Bill Gates einen Datensatz aktualisieren. In unserem relationalen Modell muss nur genau ein Datensatz geändert werden.*

## Aus der Datenbank lesen

Das Schreiben in eine Datenbank ist noch relativ einfach, das Herauslesen kann da durchaus komplexere Züge annehmen. Aber wir gehen schrittweise vor, so dass Sie auch dies vor keine unüberwindbaren Hindernisse stellen sollte. Zunächst einmal die (vereinfachte) Syntax der dazugehörigen Anweisung, SELECT:

```
SELECT Felder FROM Tabelle
 WHERE Bedingung
```

Um also alle Vor- und Nachnamen aus der Kundentabelle auszuwählen, gehen Sie wie folgt vor:

```
SELECT vorname, nachname FROM kunden
```

Alle Vor- und Nachnamen derjenigen Personen, die ein r im Nachnamen haben (Erik Frank und Linus Torvalds) erhalten Sie mit diesem Statement:

```
SELECT vorname, nachname FROM kunden
 WHERE nachname LIKE '%r%'
```

Wenn Sie alle Felder aus einer Tabelle möchten, müssen Sie nicht jeden einzelnen Spaltennamen extra angeben; verwenden Sie den Stern als Jokerzeichen:

```
SELECT * FROM kunden
```

Eine weitere Anwendung ist das Zählen der Treffer, die auf so eine Selektion gefunden werden. Hierzu dient COUNT:

```
SELECT COUNT(nachname) FROM kunden
 WHERE nachname LIKE '%r%'
```

Obige SQL-Anweisung liefert also die Anzahl der Kunden zurück, deren Nachname ein r enthält. Bei unseren Beispieldaten sind das 2 Stück.

Schon etwas schwieriger wird es, wenn aus mehreren Tabellen gleichzeitig gelesen werden soll. Dies kommt relativ oft vor, denn genau das ist ja der Witz eines relationalen Datenbankmodells. Sie müssen hier hinter FROM die Namen aller beteiligten Tabellen angeben. Vor jedem Spaltennamen sollten Sie den Namen der dazugehörigen Tabelle angeben. Dies ist zwar nur dann ein Muss, wenn ein Spaltennamen mehrfach vorkommt (in verschiedenen Tabellen), aber es dient der Verständlichkeit der Anweisung, wenn Sie so vorgehen.

Zunächst einmal soll festgestellt werden, wie viele Bestellungen Bill Gates bis dato getätigt hat. Folgende SELECT-Anweisung löst dieses Problem:

```
SELECT COUNT(bestellungen.auftragsnummer) FROM bestellungen, kunden
WHERE kunden.nachname = 'Gates'
 AND kunden.kundennummer = bestellungen.kundennummer
```

Zur Erläuterung: Es werden die Einträge in der Spalte *auftragsnummer* der Tabelle *bestellungen* gezählt. Als Bedingung muss gelten, dass der Nachname des Kunden auf »Gates« lautet. Die dazugehörige Kundennummer muss im selben Eintrag in der Tabelle *bestellungen* stehen. So ist gewährleistet, dass nur diejenigen Datensätze betrachtet werden, die zu Bill Gates gehören.

Je nachdem, welches Datenbanksystem Sie verwenden, wird nach so einer Abfrage ungefähr folgendes Ergebnis erscheinen:



```
COUNT(bestellungen.auftragsnummer)

2
(1 row(s) affected)
```

Bill Gates hat also bereits zweimal bei unserem Online-Shop bestellt.



*Sie können die SQL-Anweisung wie oben erläutert auch ein wenig kürzer fassen, indem Sie die Tabellennamen vor den Spaltennamen in den Fällen weglassen, in denen der Spaltenname eindeutig ist:*

```
SELECT COUNT(auftragsnummer) FROM bestellungen, kunden
WHERE nachname = 'Gates'
 AND kunden.kundennummer = bestellungen.kundennummer
```

Nun zu einem etwas komplexeren Beispiel: Es sollen die Bestellnummern aller Artikel ausgegeben werden, die Bill Gates bis dato bestellt hat. Der bequeme Weg sieht nun so aus, dass der Datensatz von Bill Gates gesucht wird, die entsprechende Kundennummer in einer Variablen gespeichert wird und dann mithilfe dieser Kundennummer die Tabelle *bestellungen* durchsucht wird. Anhand der dort gefundenen

Auftragsnummern kann dann über *artikel\_bestellungen* auf die Bestellnummern zugegriffen werden. Mit SQL geht das in nur einer Anweisung:

```
SELECT artikel_bestellungen.bestellnummer
FROM bestellungen, kunden, artikel_bestellungen
WHERE nachname = 'Gates'
 AND kunden.kundennummer = bestellungen.kundennummer
 AND bestellungen.auftragsnummer =
 artikel_bestellungen.auftragsnummer
```

Sie sollten eine Ausgabe in diesem Stile erhalten:



```
bestellnummer

3-8272-6003-5
3-8272-5764-6
3-8272-6003-5
3-8272-6003-5
(4 row(s) affected)
```

Wenn Sie nur an unterschiedlichen Bestellnummern interessiert sind, müssen Sie das Schlüsselwort **DISTINCT** verwenden:

```
SELECT DISTINCT artikel_bestellungen.bestellnummer
FROM bestellungen, kunden, artikel_bestellungen
WHERE nachname = 'Gates'
 AND kunden.kundennummer = bestellungen.kundennummer
 AND bestellungen.auftragsnummer =
 artikel_bestellungen.auftragsnummer
```

Hier die Ausgabe dieser Abfrage:



```
bestellnummer

3-8272-6003-5
3-8272-5764-6
(2 row(s) affected)
```

Wenn Sie das ganze auch noch lexikalisch sortiert haben möchten, müssen Sie die Anweisung **ORDER BY** verwenden. Dahinter kommt der Name der zu sortierenden Spalte (oder Spalten), und **ASC** für aufsteigend (ascending, Standard) bzw. **DESC** für absteigend (descending).

```

SELECT DISTINCT artikel_bestellungen.bestellnummer
FROM bestellungen, kunden, artikel_bestellungen
WHERE nachname = 'Gates'
 AND kunden.kundennummer = bestellungen.kundennummer
 AND bestellungen.auftragsnummer =
 artikel_bestellungen.auftragsnummer
ORDER BY artikel_bestellungen.bestellnummer

```

Die Ausgabe ändert sich wie Folgt:



```

bestellnummer

3-8272-5764-6
3-8272-6003-5
(2 row(s) affected)

```

Wenn Sie nun nicht die Bestellnummern haben möchten, sondern die Titel der Bücher, müssen Sie die Abfrage noch um die Tabelle *artikel* erweitern:

```

SELECT DISTINCT artikel.beschreibung
FROM bestellungen, kunden, artikel_bestellungen, artikel
WHERE nachname = 'Gates'
 AND kunden.kundennummer = bestellungen.kundennummer
 AND bestellungen.auftragsnummer =
 artikel_bestellungen.auftragsnummer
 AND artikel_bestellungen.bestellnummer = artikel.bestellnummer
ORDER BY artikel.beschreibung

```

Es werden die beiden Titel der von Bill Gates geordneten Bücher ausgegeben:



```

beschreibung

Louis/Wenz: Dynamic Web-Publishing in 21 Tagen
Wenz/Hauser: Jetzt lerne ich Dynamic Web-Publishing
(2 row(s) affected)

```



*Wie versprochen, hier noch ein Hinweis, wie das Auslesen eines Autowerts nach dem Einfügen eines Datensatzes programmiertechnisch zumeist gelöst wird. Einige Datenbanken bieten dazu ein eigenes SQL-Statement an, aber eben nicht alle. Man behilft sich hier insoweit, als dass man im Programmiercode eine zufällige Zeichenkette bestimmt (beispielsweise die aktuelle Uhrzeit und ein paar Zufallszeichen). Diese Zeichenkette wird in eine Spalte des neuen Datensatzes eingefügt. Sie können dann gezielt nach dieser Spalte suchen (denn die zufällige Zeichenkette ist eindeutig), und somit den Autowert ermitteln.*

## Daten löschen

Manche Leute vertreten die Lehrmeinung, dass man aus einer Datenbank nie Daten löschen sollte, sondern immer eine eigene Spalte haben sollte, welche bei einer Löschung mit einem bestimmten Wert beschrieben wird. Wie dem auch sei, schon alleine der Vollständigkeit halber müssen wir auch auf das Löschen von Daten eingehen. Hierzu dient die SQL-Anweisung DELETE:

```
DELETE FROM Tabelle
 WHERE Bedingung
```

Um also den Kunden Bill Gates vollständig zu löschen, würden Sie folgendes Kommando verwenden:

```
DELETE FROM kunden
 WHERE nachname = 'Gates'
```

Wie verwenden den Konjunktiv - *würden* -, denn Sie sollten dieses Kommando an dieser Stelle nicht ausprobieren, da Sie sich sonst Ihre schönen Testdaten korrumpieren würden.

Wenn Sie eine ganze Tabelle löschen möchten, verwenden Sie DROP TABLE:

```
DROP TABLE kunden
```

Soweit unsere Schnelleinführung in SQL. SQL bietet mannigfaltige Möglichkeiten, und alleine aus Platzgründen können wir nicht auf alle eingehen. Für die Anwendungen, die Sie am heutigen Tage noch erstellen werden, ist Ihr Wissen jetzt jedoch ausreichend.

Je nach verwendetem Datenbanksystem gibt es eine andere Maske, in die Sie diese Abfragen eingeben können. Wir gehen hier nicht en detail darauf ein, denn es geht uns heute darum, mit Skriptsprachen eine Datenbank anzusteuern. Bitte konsultieren Sie hier die Dokumentation zu Ihrer Datenbank, wenn Sie diese Schritte von Hand durchführen möchten. In Abbildung 20.1 sehen Sie exemplarisch den MySqlManager, der bei MySQL dabei ist.

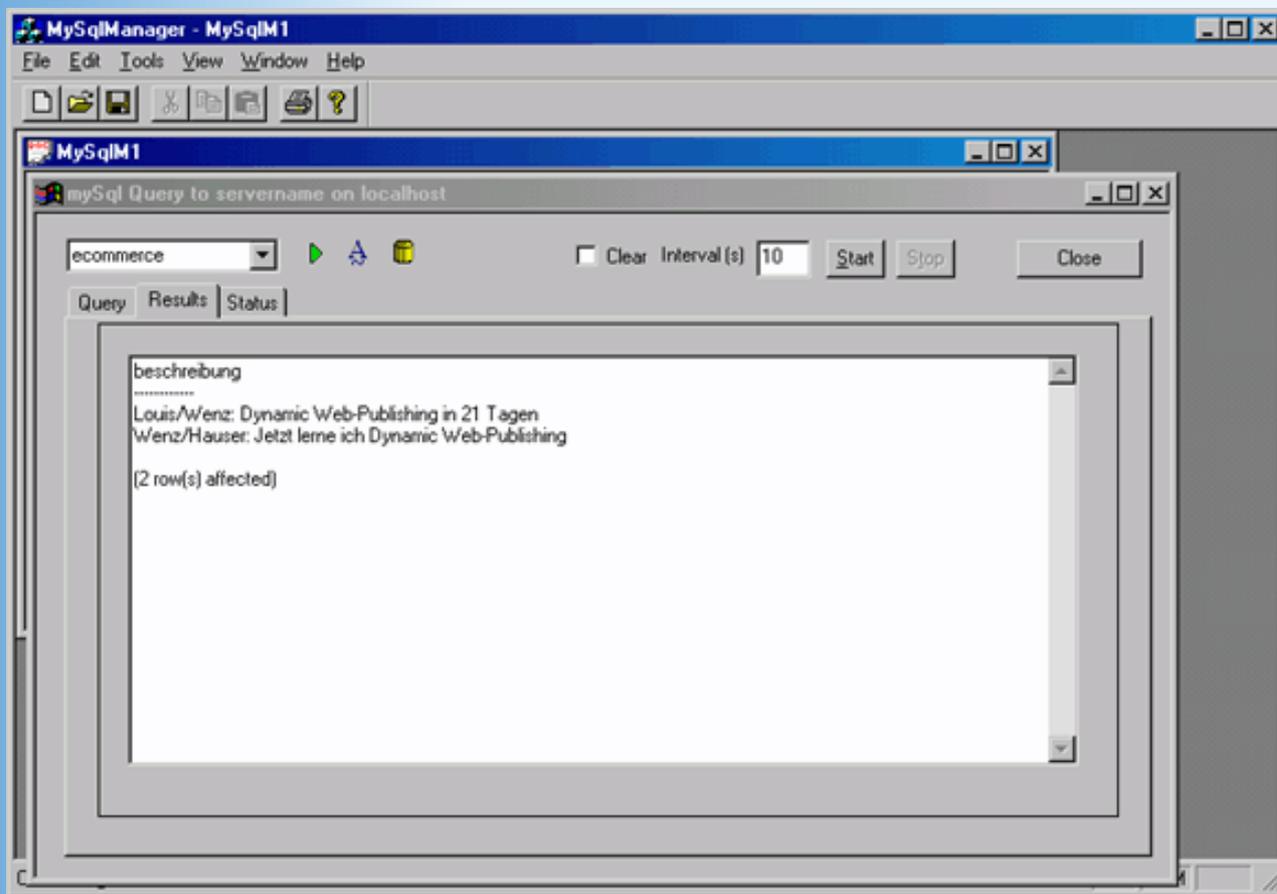


Abbildung 20.1: Der MySQLManager von MySQL für Windows

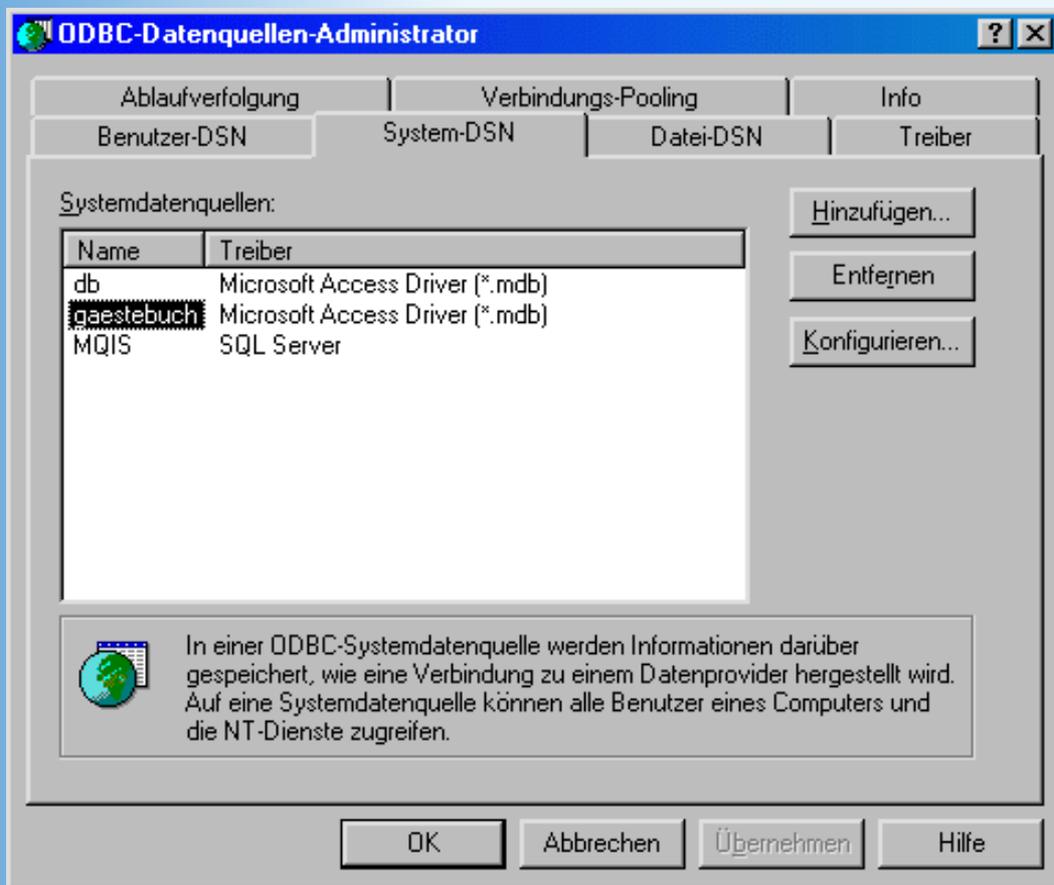
## 20.3 ODBC

Unter *Open Database Connectivity*, oder auch kurz ODBC, versteht man den Standard des Datenbankzugriffs unter Windows. Das Prinzip ist genauso einfach wie überzeugend. Es ist egal, ob Sie als Datenbank Access, MySQL, SQL Server, Oracle oder gar Excel einsetzen. Zu jeder Datenbank muss einfach ein ODBC-Treiber installiert sein, der die Hauptarbeit erledigt. Sie, in Ihrer Rolle als Skriptprogrammierer, greifen dann über ODBC auf die Datenbank zu, und schicken SQL-Abfragen an den ODBC-Treiber. Dieser Treiber wandelt dann die SQL-Kommandos in die entsprechenden Abfragebefehle für die jeweilige Datenbank um, und liefert das Ergebnis der Abfrage an Ihr Skript zurück.

Ein klassisches Beispiel: Wie oben bereits erwähnt, verwendet Access als Jokerzeichen \* und ?, SQL-Standard ist jedoch % und \_. Es wäre nun ungünstig, eine Anwendung auf Access aufzusetzen, dabei \* und ? zu verwenden, und dann festzustellen, dass aufgrund der Datenmengen ein Microsoft SQL Server die bessere Lösung wäre. Ohne ODBC müssten Sie nun viele Ihrer Abfragen umschreiben, bei der Verwendung von ODBC müssen Sie nichts ändern.

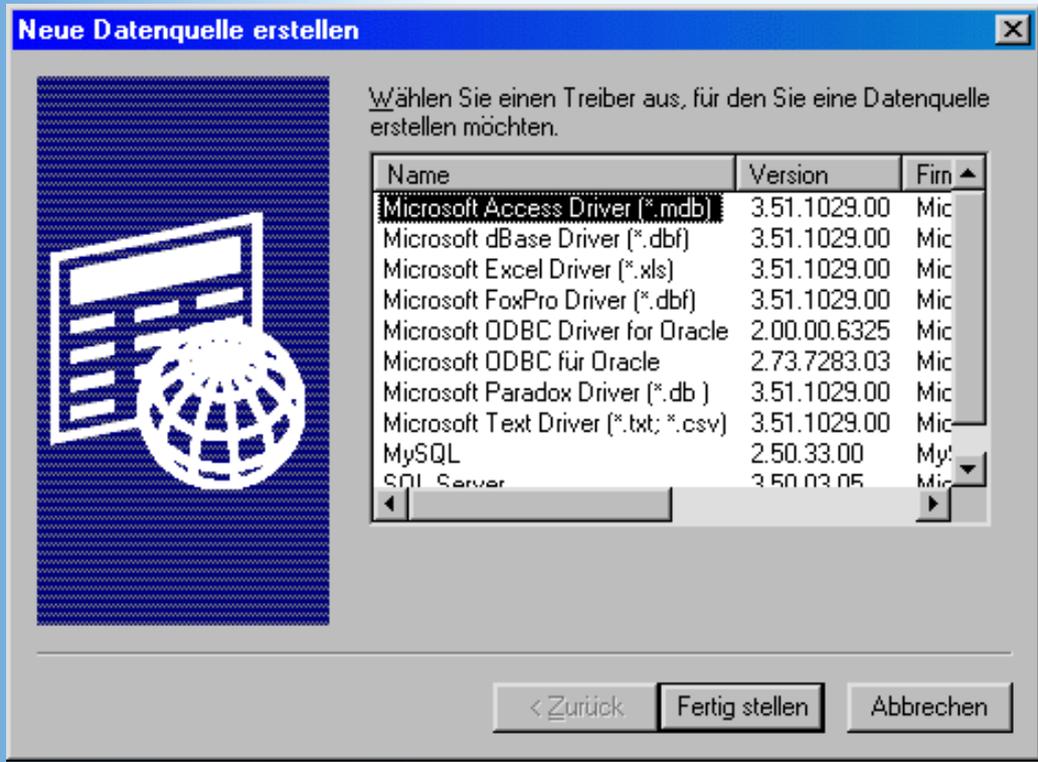
Beim ODBC-Zugriff taucht häufig der Begriff *DSN* auf. Das steht für *Data Source Name*, und ist der Name, über den die Datenbank vom ODBC-Treiber aus angesprochen werden kann. Stellen Sie sich das einfach so vor: Der Treiber ruft die Datenbank über ihren Spitznamen auf. Wenn Sie nun von einer Datenbank auf die andere umsteigen, müssen Sie nur der neuen Datenbank den Spitznamen der alten Datenbank zuweisen, und Ihre Skripte funktionieren weiterhin tadellos.

Einen DSN-Eintrag können Sie über die Systemsteuerung (**Start/Einstellungen/ Systemsteuerung**) einrichten. Je nach Betriebssystemversion heißt der entsprechende Eintrag »Datenquellen 32-Bit«, »ODBC-Datenquellen (32-Bit)« oder ähnlich. Klicken Sie zunächst doppelt auf das Symbol in der Systemsteuerung, und klicken Sie dann auf das Register **System-DSN**. Das sieht dann ungefähr so aus wie in Abbildung 20.2.



**Abbildung 20.2: Das Register System-DSN**

Wählen Sie nun die Schaltfläche **Hinzufügen**, und es erscheint eine Liste aller im System zur Verfügung stehenden ODBC-Treiber (siehe Abbildung 20.3).



**Abbildung 20.3: Die ODBC-Treiber, die im System installiert sind**

Wählen Sie nun einen der Treiber aus, und klicken Sie auf **Fertig stellen**. Es erscheint nun eine

Eingabemaske, die je nach ODBC-Treiber unterschiedlich ist. Dort können Sie die entsprechende Datenbank auswählen oder sogar erstellen lassen. In Abbildung 20.4 sehen Sie, was beim ODBC-Treiber für MySQL zur Auswahl steht; in Abbildung 20.5 das entsprechende Dialogfenster für Access. Das wichtigste Feld ist auf jeden Fall das für den DSN-Namen; unter diesem Namen (im Beispiel: *umfrage*) können Sie von Ihren Skripten aus die Datenbank ansprechen.

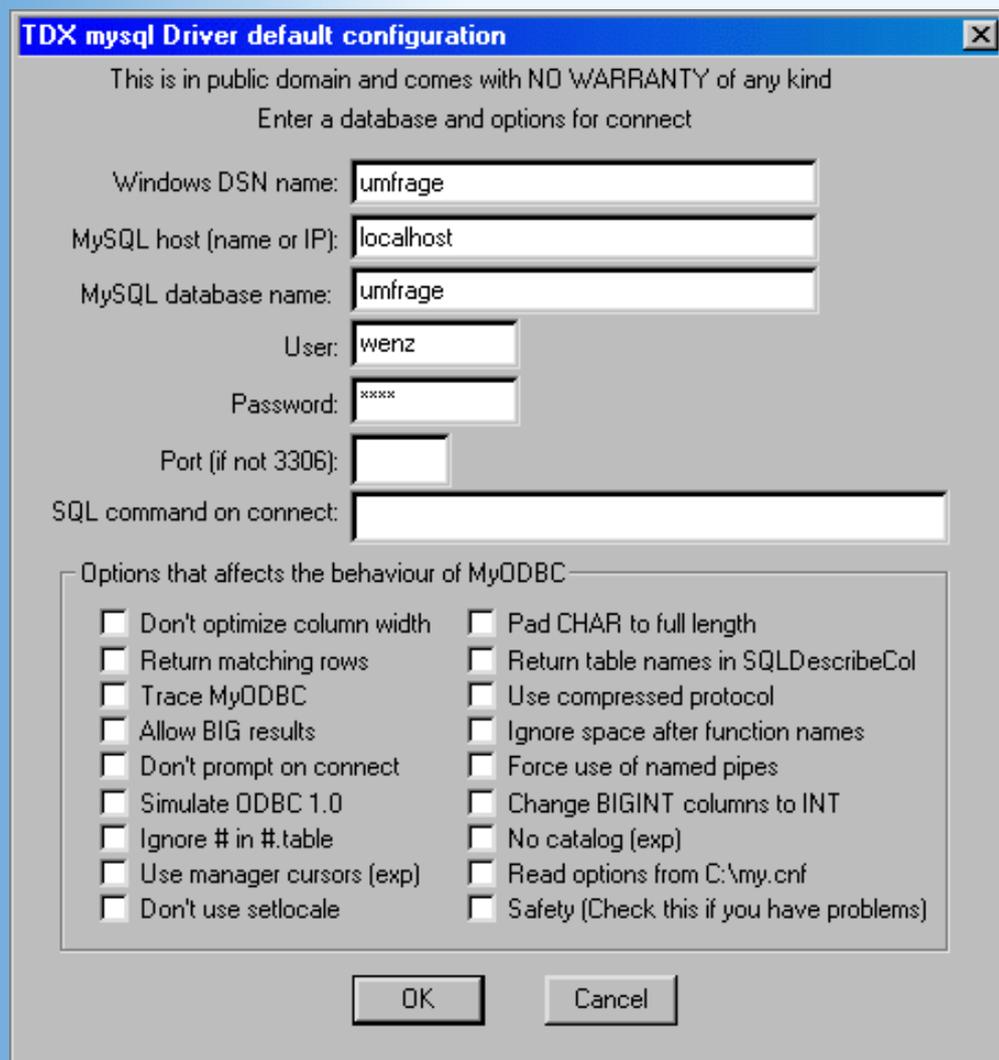
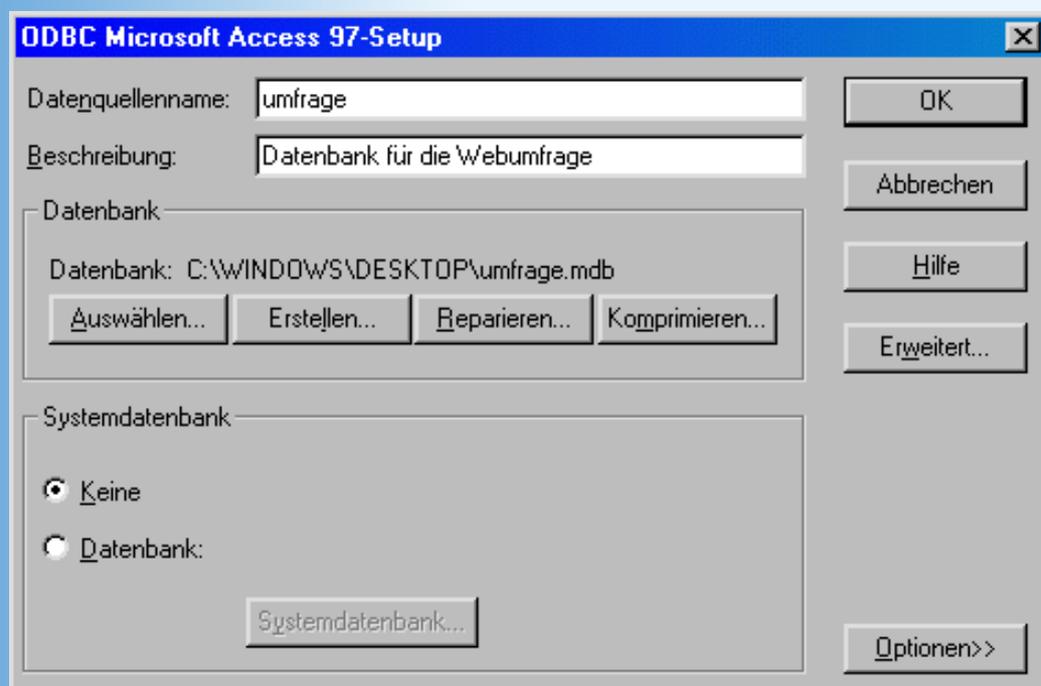


Abbildung 20.4: Die Einstellungen von MyODBC, dem ODBC-Treiber für MySQL



## Abbildung 20.5: Die Einstellungen im ODBC-Treiber für Microsoft Access

Nach erfolgreicher Einstellung erscheint der neu eingerichtete DSN-Name in der Auflistung im Register System-DSN, und Sie sind startbereit.

## 20.4 Datenbanken mit ASP

Im Folgenden führen wir Sie in die Geheimnisse der Datenbankprogrammierung mit ASP ein. Als Beispiel werden wir das Ihnen bereits bekannte Gästebuch verwenden. Wir werden sowohl in die Datenbank schreiben, als auch Daten aus der Datenbank auslesen, oder neudeutsch »eine Auswertung fahren«.

### Datenbankzugriff

Unter ASP ist die ActiveX-Komponente ADODB zuständig. Das ist im übrigen die Kurzform für ActiveX Data Objects Database.

Zunächst einmal müssen Sie eine Verbindung zur Datenbank herstellen. Hierzu benötigen Sie ein Objekt vom Typ ADODB.Connection:

```
<%
Set objConn = Server.CreateObject("ADODB.Connection")
%>
```

Dann müssen Sie diesem Objekt die notwendigen Parameter übergeben: Zu welcher Datenbank soll die Komponente sich verbinden, und wie lautet gegebenenfalls der Benutzername und das Passwort für den Datenbankzugriff:

```
<%
Conn.Open "dsn=gaestebuch;database=gaestebuch;uid=christian;pwd=geheim"
%>
```



*Zwei Hinweise: Wenn Sie Ihre Datenbank nicht mit einem Passwort schützen, lassen Sie uid und pwd einfach weg. Und, wenn Sie Access verwenden, benötigen Sie unter Umständen auch keinen Benutzernamen, lassen Sie hier den uid-Abschnitt weg.*

Wenn Sie keine DSN einrichten möchten oder können, gibt es bei Access auch noch die Möglichkeit, ohne DSN-Eintrag auszukommen. Dazu müssen Sie den kompletten Pfad der Datenbank angeben:

```
<%
Conn.Open "DRIVER={Microsoft Access Driver (*.mdb)};" & _
"DBQ=c:\datenbanken\gaestebuch.mdb;"
%>
```

Sie können nun SQL-Statements an die Datenbankverbindung schicken, indem Sie die Methode Execute aufrufen:

```
<%
objConn.Execute "INSERT INTO gaestebuch (name, kommentar) " &_
"VALUES ('Christian', 'ganz klasse!') "
%>
```

Wenn Sie ein SELECT-Statement ausführen und am Rückgabewert interessiert sind, benötigen Sie ein weiteres Objekt, das eine Ergebnisliste darstellt. Der Name des Objekts ist ADODB.RecordSet, Sie müssen das aber nicht explizit mit Server.CreateObject deklarieren, das macht der ASP-Interpreter automatisch für Sie:

```
<%
Set objRs = objConn.Execute("SELECT * FROM gaestebuch")
%>
```

Das Recordset-Objekt verhält sich wie ein assoziatives Array: Wie bei beispielsweise PHP können Sie über den Schlüssel (in diesem Fall: den Spaltennamen) auf den dazugehörigen Wert zugreifen:

```
<%
Response.Write objRs("name")
%>
```



*Alternativ dazu können Sie auch über die Position auf die einzelnen Werte zugreifen: mit objRs(0) erhalten Sie den Wert der ersten Spalte, mit objRs(1) den der zweiten Spalte, und so weiter.*

Eine Methode und eine Eigenschaft des Recordset-Objekts verdienen noch besondere Erwähnung: Mit objRs.MoveNext wird der nächste Eintrag in einer Ergebnisliste angesprungen; und objRs.EOF gibt an, ob der letzte Eintrag schon erreicht worden ist oder nicht. Folgender Beispielcode gibt beispielsweise alle Kommentare aus, die sich in der Datenbank gaestebuch befinden:

```
<%
Set objConn = Server.CreateObject("ADODB.Connection")
objConn.Open "dsn=gaestebuch;database=gaestebuch "
Set objRs = objConn.Execute("SELECT * FROM gaestebuch")
While Not objRs.EOF
 Response.Write objRs("kommentar") & "
"
 objRs.MoveNext
Wend
Set objRs = Nothing
objConn.Close
Set objConn = Nothing
%>
```

Solange der Interpreter noch nicht am Ende der Ergebnisliste angekommen ist (objRs.EOF wäre dann True), wird der Kommentar im aktuellen Eintrag in der Ergebnisliste ausgegeben und dann der nächste Eintrag angesprungen.



*Beachten Sie auch, dass am Ende des Skripts alle Objekte wieder aus dem Speicher gelöscht werden, indem sie auf Nothing gesetzt werden!*

## Gästebuch

Erinnern Sie sich noch an das Gästebuch von Tag 3? Dort befand sich - mit leichten Änderungen - der folgende HTML-Code:

### Listing 20.9: gaestebuch.html

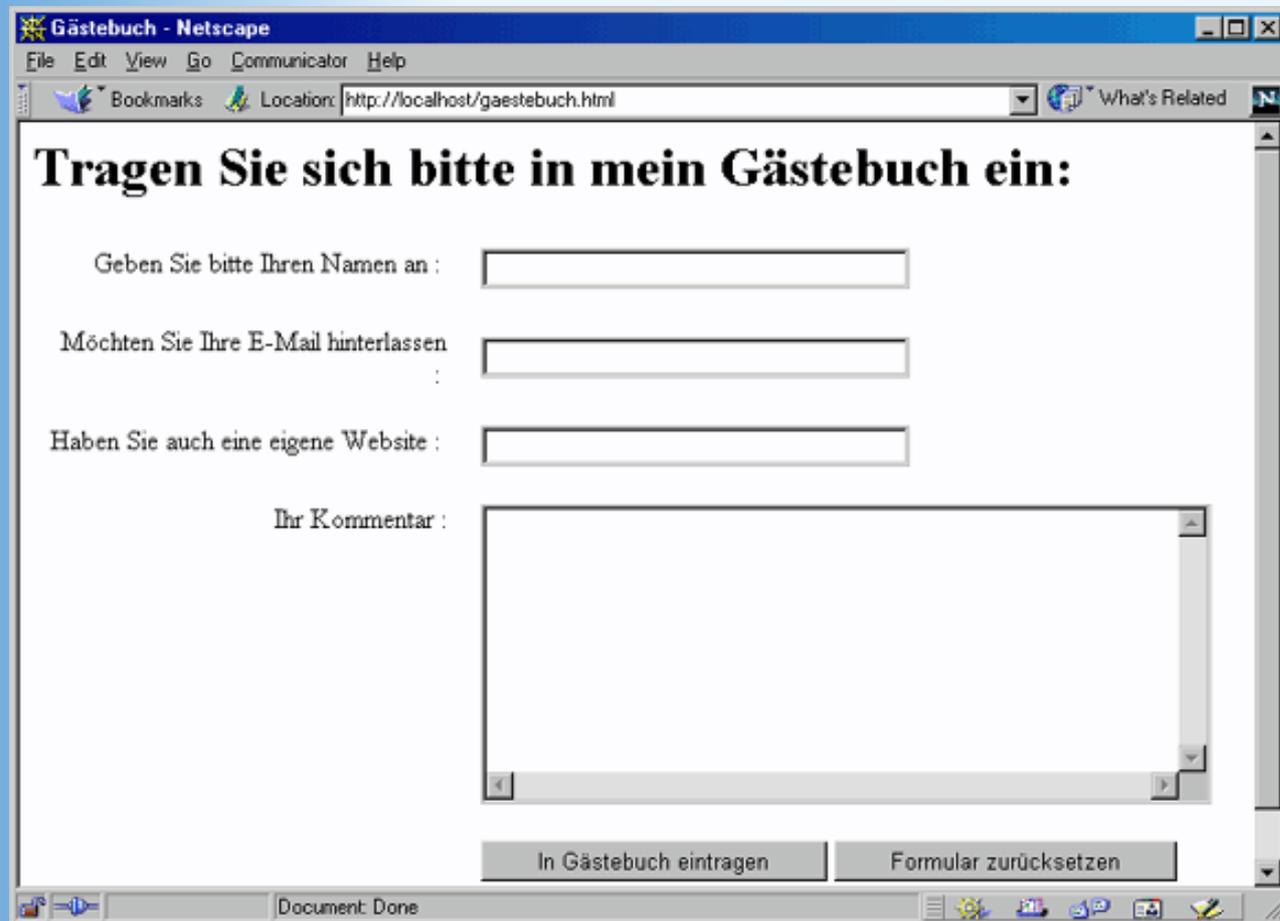
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Gästebuch</title>
</head>
<body bgcolor="white">
<h1>Tragen Sie sich bitte in mein Gästebuch ein:</h1>
<form action="gaestebuch_eintragen.asp" method="post">
 <table border="0" cellspacing="0" cellpadding="10">
 <colgroup span=2">
 <col width="230">
 <col width="450">
 </colgroup>
 <tr>
 <td align="right" valign="top" width="230">
 Geben Sie bitte Ihren Namen an : </td>
 <td><input name="name" size="30" maxlength="50"></td>
 </tr>
 <tr>
 <td align="right" valign="top" width="230">
 Möchten Sie Ihre E-Mail hinterlassen : </td>
 <td><input name="email" size="30" maxlength="50"></td>
 </tr>
 <tr>
 <td align="right" valign="top" width="230">
 Haben Sie auch eine eigene Website : </td>
 <td><input name="website" size="30" maxlength="50"></td>
 </tr>
 <tr>
 <td align="right" valign="top" width="230">Ihr Kommentar :</td>
 <td><textarea name="kommentar" rows="9" cols="50"></textarea></td>
 </tr>
 <tr>
 <td> </td>
 <td><input type="submit" value="In Gästebuch eintragen">
 <input type="reset" value="Formular zurücksetzen"></td>
```

```

 </tr>
</table>
</form>
</body>
</html>

```

Zur Erinnerung sehen Sie in Abbildung 20.6 noch einmal, wie diese Seite im Browser aussieht.



**Abbildung 20.6: Das Gästebuch im Browser**

Beachten Sie, wie wir das action-Attribut des Formulars auf ein Skript namens *gaestebuch\_eintragen.asp* verwiesen haben. Diese Datei *gaestebuch\_eintragen.asp* muss die Werte aus dem Formular nehmen und in die Datenbank schreiben. Die Datenbank selbst ist sehr einfach eingebaut: Für Name, E-Mail-Adresse, Website und Kommentar wird jeweils ein Textfeld definiert.

Das Kommentarfeld sollte möglichst viele Zeichen haben (je nach Datenbank beispielsweise 255 oder 8000), bei den anderen Feldern reichen jeweils 50 Zeichen. Das Schreiben wird dann wie folgt realisiert:

**Listing 20.10: gaestebuch\_eintragen.asp**

```

<%
Set objConn = Server.CreateObject("ADODB.Connection")
objConn.Open "dsn=gaestebuch;database=gaestebuch "
strSQL = "INSERT INTO gaestebuch (name, email, web, kommentar) " & _
 "VALUES ('" & Request.Form("name") & "', " & _
 "'" & Request.Form("email") & "', " & _
 "'" & Request.Form("website") & "', " & _
 "'" & Request.Form("kommentar") & "'")"

```

```

objConn.Execute(strSQL)
objConn.Close
Set objConn = Nothing
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Gästebuch</title>
</head>
<body bgcolor="white">
<h1>Daten wurden ins Gästebuch geschrieben!</h1>
<p>Zum Gästebuch</p>
</body>
</html>

```

Das Skript *gaestebuch\_zeigen.asp*, auf das verwiesen wird, liest wiederum alle Einträge aus dem Gästebuch aus und zeigt sie im Webbrowser an:

### Listing 20.11: gaestebuch\_zeigen.asp

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Gästebuch</title>
</head>
<body bgcolor="white">
<h1>Einträge im Gästebuch:</h1>
<%
Set objConn = Server.CreateObject("ADODB.Connection")
objConn.Open "dsn=gaestebuch;database=gaestebuch "
Set objRs = objConn.Execute("SELECT * FROM gaestebuch")
While Not objRs.EOF
 Response.Write "<p>Name: " & _
 Server.HTMLEncode("" & objRs("kommentar")) & "
"
 Response.Write "E-Mail: " & _
 Server.HTMLEncode("" & objRs("email")) & "
"
 Response.Write "Website: " & _
 Server.HTMLEncode("" & objRs("web")) & "
"
 Response.Write "Kommentar: " & _
 Server.HTMLEncode("" & objRs("kommentar")) & "</p>>"
 objRs.MoveNext
 Response.Write "<hr />"
Wend
Set objRs = Nothing
objConn.Close
Set objConn = Nothing
%>
</body>
</html>

```

Wozu dient `Server.HTMLEncode("" & objRs("Spaltenname"))`? Nun, `Server.HTMLEncode` wandelt einen

String in das entsprechende HTML-Äquivalent um. Dies verhindert, das böswillige Zeitgenossen HTML-Code in den Postings verwenden und somit beispielsweise obszöne Grafiken einfügen können. Wenn aber nun objRs("Spaltenname") leer ist, und Sie darauf Server.HTMLEncode aufrufen, gibt es eine Fehlermeldung. Sie müssen also zunächst durch Konkatenation mit einem Leerstring diesen Nullwert in einen String umwandeln.

Somit haben Sie das nötige Rüstwerkzeug, um mit ASP eigene Datenbanken zu programmieren. Wenn Sie eher ein Fan von PHP sind (oder Sie keine andere Wahl haben, Unix/Linux sei Dank), lesen Sie den folgenden Abschnitt!

## 20.5 Datenbanken mit PHP

Wenn Sie PHP einsetzen, haben Sie in Bezug auf Datenbanken die Qual der Wahl. Außer den Klassikern MySQL und ODBC unterstützt PHP eine ganze Reihe von weiteren Datenbanken direkt, etwa Microsoft SQL Server oder Oracle. In der Praxis werden Sie aber entweder auf MySQL setzen, oder eine Datenbank verwenden, die Sie mit ODBC ansteuern. Aus diesem Grund beschränken wir uns auch hier auf diese beiden Typen.

### Datenbankzugriff

Prinzipiell funktioniert der Zugriff unter PHP ähnlich wie bei ASP. Zunächst müssen Sie eine Verbindung zu einer Datenbank aufbauen, dann ein SQL-Kommando an die Datenbank schicken, und eventuell eine Ergebnisliste auswerten. Der Teufel steckt - wie immer - im Detail.

### MySQL

Die Verbindung zum MySQL-Server geschieht mit folgendem Kommando:

```
<?php
 mysql_connect("localhost", "benutzer", "geheim");
?>
```

Anstelle von localhost müssen Sie den Namen des MySQL-Servers eingeben. Sollte dieser lokal auf derselben Maschine wie der Webserver laufen, genügt localhost.

Als nächstes müssen Sie eine Datenbank auswählen, auf die Sie zugreifen möchten:

```
<?php
 mysql_select_db("gaestebuch");
?>
```

Um Daten in die Datenbank zu schreiben, reicht es, ein entsprechendes SQL-Statement an die MySQL-Datenbank zu schicken. Das entsprechende Kommando heißt `mysql_query`.

```
<?php
 $sql = "INSERT INTO gaestebuch (name,email,web,kommentar) VALUES (";
 $sql .= "' ' . $HTTP_POST_VARS["name"] . "', ";
 $sql .= "' ' . $HTTP_POST_VARS["email"] . "', ";
 $sql .= "' ' . $HTTP_POST_VARS["website"] . "', ";
 $sql .= "' ' . $HTTP_POST_VARS["kommentar"] . "')";
 mysql_query($sql);
?>
```

Am Ende sollten Sie die Verbindung zur Datenbank wieder schließen, um den Ressourcenverbrauch gering zu halten:

```
<?php
 mysql_close();
?>
```

Wenn Sie Daten aus der Datenbank auslesen möchten, müssen Sie ein paar Zeilen mehr programmieren. Die Funktion `mysql_query` gibt eine numerische ID zurück, welche als Handle für die Ergebnisliste dient. In anderen Worten: Über diese ID können Sie auf die Ergebnisliste zugreifen. Die Funktion `mysql_fetch_array` liefert ein assoziatives Array mit allen Werten im aktuellen Eintrag in der Ergebnisliste zurück. Gleichzeitig wird das nächste Element der Ergebnisliste angesprungen, so dass ein erneuter Aufruf von `mysql_fetch_array` den nächsten Eintrag zurückliefert. Um also alle Kommentare in der Datenbank auszugeben, ist dieser Code hilfreich:

```
<?php
 mysql_connect("localhost", "benutzer", "geheim");
 mysql_select_db("gaestebuch");
 $sid = mysql_query("SELECT * FROM gaestebuch");
 while ($eintrag = mysql_fetch_array($sid)) {
 print($eintrag["kommentar"] . "
");
 }
 mysql_close();
?>
```

## ODBC

Wenn Sie ODBC einsetzen möchten, müssen Sie bei PHP auf jeden Fall eine DSN einrichten, es funktioniert hier also (noch) nicht mit Angabe der Access-Datei. Aber da PHP permanent weiterentwickelt wird, ist es durchaus möglich, dass sich das einmal ändern wird.

Aber nun zur grauen Theorie: Die erste Funktion, die Sie verwenden müssen, ist `odbc_connect`, welche eine Verbindung zur Datenbank aufbaut. Alle drei Parameter - DSN, Benutzername und Passwort - sind Pflicht. Wenn Sie Ihre Datenbank nicht geschützt haben, können Sie als Benutzernamen und Passwort jeweils einen Leerstring angeben. Die Funktion gibt eine ID zurück, welche Sie später noch benötigen (Sie können also parallel mehrere Verbindungen öffnen, und anhand der IDs können Sie später angeben, welches SQL-Statement an welche Verbindung und damit an welche Datenbank übergeben wird).

```
<?php
 $sid = odbc_connect("gaestebuch", "benutzer", "geheim");
?>
```

Um nun ein SQL-Kommando an die Datenbank zu übergeben, rufen Sie `odbc_exec` auf. Als Parameter übergeben Sie außer dem numerischen Handle das entsprechende SQL-Statement:

```
<?php
 $sql = "INSERT INTO gaestebuch (name,email,web,kommentar) VALUES (";
 $sql .= "' ' . $HTTP_POST_VARS["name"] . "', ";
 $sql .= "' ' . $HTTP_POST_VARS["email"] . "', ";
 $sql .= "' ' . $HTTP_POST_VARS["website"] . "', ";
 $sql .= "' ' . $HTTP_POST_VARS["kommentar"] . "')";
 odbc_exec($sid, $sql);
```

```
?>
```

Am Ende sollten Sie die Verbindung zur Datenbank noch schließen:

```
<?php
 odbc_close($id);
?>
```

Wenn Sie nun Daten aus der Datenbank auslesen möchten, müssen Sie den Rückgabewert von `odbc_exec` in einer Variablen zwischenspeichern. Sie ahnen es bereits - auch dieser Rückgabewert ist eine numerische ID, und kann später zur Abfrage der Ergebnisliste verwendet werden. Was bei MySQL noch mit `mysql_fetch_array` ging, funktioniert bei ODBC mit `odbc_fetch_row`. Diese Funktion liefert aber nicht ein assoziatives Array mit den Einträgen des aktuellen Ergebnisses zurück, sondern liest das aktuelle Ergebnis in den Speicher. Auf die einzelnen Einträge zugreifen können Sie dann mit `odbc_result`. Als Parameter übergeben Sie - neben der ID der Abfrage - noch den gewünschten Spaltennamen.

Im folgenden Beispiel werden wieder einmal alle Kommentare in der Datenbank untereinander ausgegeben:

```
<?php
 $id = odbc_connect("gaestebuch", "benutzer", "geheim");
 $eintrag = odbc_exec($id, "SELECT * FROM gaestebuch");
 while (odbc_fetch_row($eintrag)) {
 print(odbc_result($eintrag, "kommentar") . "
");
 }
 odbc_close($id);
?>
```

## Gästebuch

Kommen wir nun zum Gästebuch. Egal, ob Sie MySQL oder ODBC einsetzen, Sie müssen zunächst das `action`-Attribut im Gästebuch auf eine entsprechende URL anpassen:

### Listing 20.12: gaestebuch\_php.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Gästebuch</title>
</head>
<body bgcolor="white">
<h1>Tragen Sie sich bitte in mein Gästebuch ein:</h1>
<form action="gaestebuch_eintragen.php" method="post">
 <table border="0" cellspacing="0" cellpadding="10">
 <colgroup span=2">
 <col width="230">
 <col width="450">
 </colgroup>
 <tr>
 <td align="right" valign="top" width="230">
 Geben Sie bitte Ihren Namen an
 : </td>
```

```

 <td><input name="name" size="30" maxlength="50"></td>
</tr>
<tr>
 <td align="right" valign="top" width="230">
 Möchten Sie Ihre E-Mail hinterlassen : </td>
 <td><input name="email" size="30" maxlength="50"></td>
</tr>
<tr>
 <td align="right" valign="top" width="230">
 Haben Sie auch eine eigene Website : </td>
 <td><input name="website" size="30" maxlength="50"></td>
</tr>
<tr>
 <td align="right" valign="top" width="230">Ihr Kommentar :</td>
 <td><textarea name="kommentar" rows="9" cols="50"></textarea></td>
</tr>
<tr>
 <td> </td>
 <td><input type="submit" value="In Gästebuch eintragen">
 <input type="reset" value="Formular zurücksetzen"></td>
</tr>
</table>
</form>
</body>
</html>

```

Das Schreiben in die Datenbank mit PHP (Skript *gaestebuch\_eintragen.php*) funktioniert nun mit MySQL wie folgt:

### Listing 20.13: gaestebuch\_eintragen\_mysql.php

```

<?php
mysql_connect("localhost", "benutzer", "geheim");
mysql_select_db("gaestebuch");
$sql = "INSERT INTO gaestebuch (name,email,web,kommentar) VALUES (";
$sql .= "' . $HTTP_POST_VARS["name"] . "', ";
$sql .= "' . $HTTP_POST_VARS["email"] . "', ";
$sql .= "' . $HTTP_POST_VARS["website"] . "', ";
$sql .= "' . $HTTP_POST_VARS["kommentar"] . "')";
mysql_query($sql);
mysql_close();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Gästebuch</title>
</head>
<body bgcolor="white">
<h1>Daten wurden ins Gästebuch geschrieben!</h1>
<p>Zum Gästebuch</p>
</body>
</html>

```

Bei Verwendung von ODBC ändert sich kaum etwas, nur die Funktionen heißen anders, und Sie müssen mit dem numerischen Handle der Verbindung arbeiten:

#### Listing 20.14: gaestebuch\_eintragen\_odbc.php

```
<?php
 $id = odbc_connect("gaestebuch", "benutzer", "geheim");
 $sql = "INSERT INTO gaestebuch (name,email,web,kommentar) VALUES (";
 $sql .= "'" . $_HTTP_POST_VARS["name"] . "', ";
 $sql .= "'" . $_HTTP_POST_VARS["email"] . "', ";
 $sql .= "'" . $_HTTP_POST_VARS["website"] . "', ";
 $sql .= "'" . $_HTTP_POST_VARS["kommentar"] . "')";
 odbc_exec($id, $sql);
 odbc_close($id);
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Gästebuch</title>
</head>
<body bgcolor="white">
<h1>Daten wurden ins Gästebuch geschrieben!</h1>
<p>Zum Gästebuch</p>
</body>
</html>
```

Die Datei *gaestebuch\_zeigen.php*, auf die verwiesen wird, liest alle Daten aus der Datenbank aus und gibt sie aus. Aus oben bereits erläuterten Sicherheitsgründen werden auch hier die Einträge zunächst in ein HTML-konformes Format umgewandelt. Diesen Zweck erfüllt die PHP-Funktion `htmlspecialchars`.

Zunächst der Code für MySQL:

#### Listing 20.15: gaestebuch\_zeigen\_mysql.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Gästebuch</title>
</head>
<body bgcolor="white">
<h1>Einträge im Gästebuch:</h1>
<?php
 mysql_connect("localhost", "benutzer", "geheim");
 mysql_select_db("gaestebuch");
 $id = mysql_query("SELECT * FROM gaestebuch");
 while ($eintrag = mysql_fetch_array($id)) {
 print("<p>Name: ");
 print(htmlspecialchars($eintrag["name"]) . "
");
 print("E-Mail: ");
 print(htmlspecialchars($eintrag["email"]) . "
");
 }
?>
```

```

 print("Website: ");
 print(htmlspecialchars($eintrag["web"]) . "
");
 print("Kommentar: ");
 print(htmlspecialchars($eintrag["kommentar"]) . "</p>");
 }
 mysql_close();
?>
</body>
</html>

```

Und zu guter Letzt die Adaption für ODBC:

### Listing 20.16: gaestebuch\_zeigen\_odbc.php

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Gästebuch</title>
</head>
<body bgcolor="white">
<h1>Einträge im Gästebuch:</h1>
<?php
 $id = odbc_connect("gaestebuch", "benutzer", "geheim");
 $eintrag = odbc_exec($id, "SELECT * FROM gaestebuch");
 while (odbc_fetch_row($eintrag)) {
 print("<p>Name: ");
 print(htmlspecialchars($odbc_result($eintrag, " name"));
 print("
");
 print("E-Mail: ");
 print(htmlspecialchars($odbc_result($eintrag, "email"));
 print("
");
 print("Website: ");
 print(htmlspecialchars($odbc_result($eintrag, "web"));
 print("
");
 print("Kommentar: ");
 print(htmlspecialchars($odbc_result($eintrag, "kommentar"));
 print("</p>");
 }
 odbc_close($id);
?>
</body>
</html>

```

In Abbildung 20.7 sehen Sie eine mögliche Ausgabe des Skripts *gaestebuch\_zeigen.php* im Browser.



Abbildung 20.7: Exemplarische Einträge im Gästebuch

## 20.6 Fragen und Antworten

### Frage:

Wie kann ich feststellen, wie viele Einträge in der Ergebnisliste stehen?

### Antwort:

Verwenden Sie eine *while*-Schleife, durchschreiten Sie die Eingabeliste und zählen Sie dabei in einer Variablen mit. Wenn Sie ASP verwenden, können Sie auch `Set objRs = objConn.Execute("SELECT COUNT(*) FROM tabelle")` ausführen; die Anzahl steht dann in `objRs(0)`.

### Frage:

Wenn ich Strings, die einen Apostroph enthalten, in die Datenbank einfügen will, erhalte ich eine Fehlermeldung. Was kann ich tun?

### Antwort:

Ersetzen Sie Apostrophe (') durch doppelte Apostrophe ("). In ASP verwenden Sie dazu `strX = Replace(strX, "'", "''")`, in PHP `str_replace("'", "", $x)`.

## 20.7 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

1. Nennen Sie fünf SQL-Statements!
2. Was ist ein Autowert?
3. Nennen Sie zwei Möglichkeiten, eine Tabelle anzulegen!
4. Wozu benötigen Sie einen DSN-Eintrag?

## Übungen

1. Die Gästebucheinträge sollen nach dem Datum der Erstellung ausgegeben werden, der aktuellste Eintrag zuerst. Überlegen Sie sich zwei verschiedene Ansätze, dieses Problem zu lösen, und setzen Sie einen davon in die Praxis um.

---

❖ Kapitel Inhalt Index **SAMS** ❖ Top Kapitel ❖

---

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

## Tag 21

# XML und XHTML

In Kapitel 2 haben wir Sie mehrfach auf die XHTML-Spezifikation des W3C-Konsortiums verwiesen. Heute, quasi zum Ausklang des Buches, wollen wir Ihnen die XHTML- Spezifikation und die dahinter stehende XML-Technologie ein wenig näher vorstellen und damit gleichzeitig einen Ausblick auf das wagen, was uns die nächste Zukunft bringen wird.

Die Themen heute:

- Vorstellung von XML
- Einführung in die XML-typische Terminologie
- Erstellung und Testen einfacher XML-Dokumente
- Formatierung von XML-Dokumenten mit XSL
- Zusammenfassung der XHTML-Spezifikation

## 21.1 XML

In der ersten Hälfte des heutigen Tages wollen wir uns mit XML beschäftigen. Es war wohl im Jahre 2000, als XML eines der typischen Modethemen war. Jeder wollte auf einmal etwas mit XML anstellen, jeder wollte mitreden können. Inzwischen hat sich der Nebel ein wenig gelichtet. XML ist immer noch ein sehr wichtiges Thema, und - um eines vorwegzunehmen - ein Thema, das unserer Meinung nach große Zukunft hat. Wie bei jedem Hype hat sich aber auch ein wenig Ernüchterung breitgemacht. Der Heimanwender hat gemerkt, dass er XML (noch) nicht unbedingt einsetzen muss, zu gering ist die Unterstützung von Browserherstellern - wie gesagt, **noch**. Professionelle Anwender haben aber den Trend - und die Vorteile von XML - erkannt. Wie die Zukunft aussieht, erfahren Sie heute Mittag, zunächst aber möchten wir Ihnen eine kurze Einführung in diese (doch recht umfangreiche) Thematik geben. Als Motivation ein Praxisbeispiel, das zeigt, dass XML inzwischen auch im Produktivbetrieb eingesetzt wird. Einige Produktseiten des Computerherstellers Compaq, zu finden unter [www.compaq.de](http://www.compaq.de), sind XML-basiert.

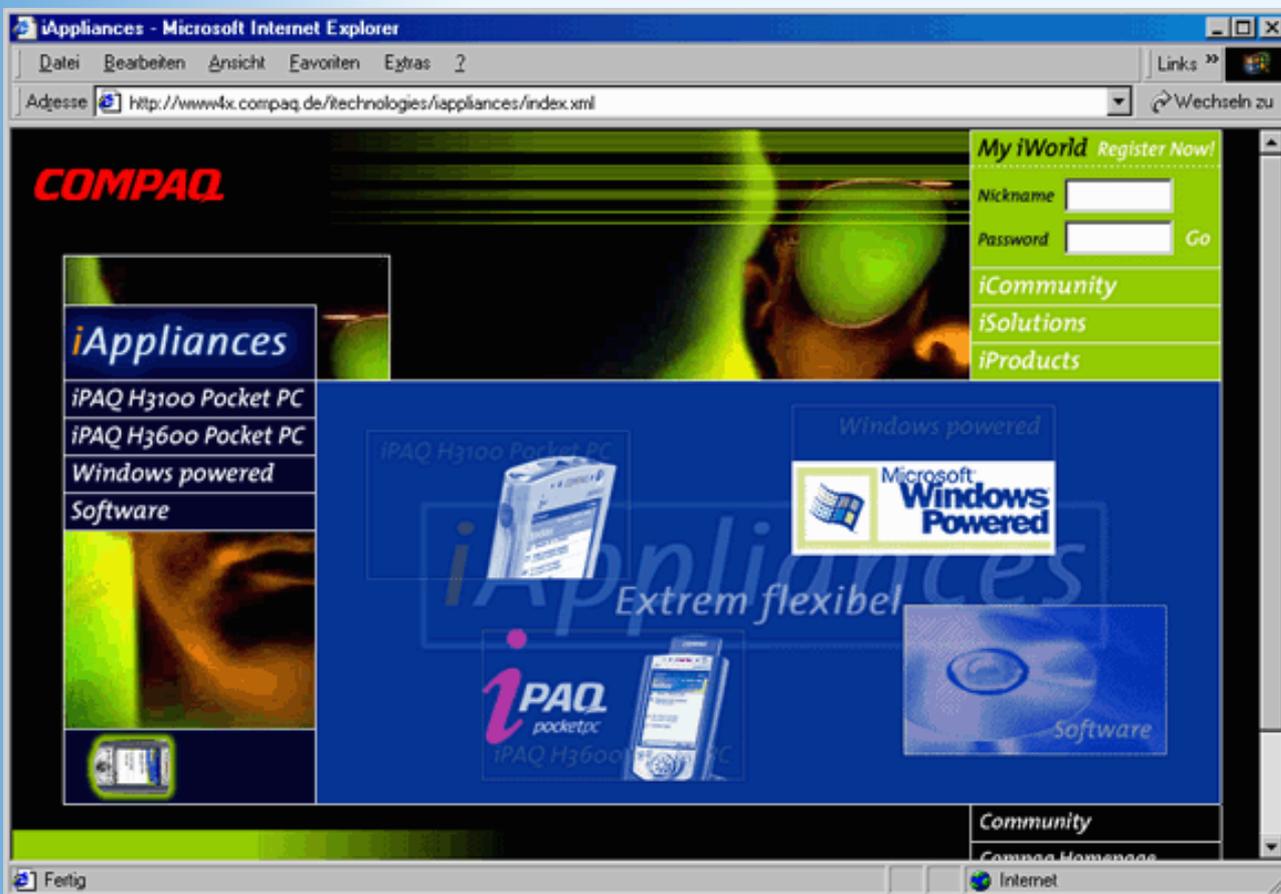


Abbildung 21.1: Eine offensichtlich XML-basierte Seite - schauen Sie auf die URL!

## Warum XML?

XML steht für eXtensible Markup Language, was in etwa mit erweiterbare Auszeichnungssprache übersetzt werden kann. Und diese Erweiterbarkeit ist auch der wohl größte Vorteil von XML gegenüber HTML. Heutzutage stößt man immer öfters an die Grenzen von HTML. Mühsam wird eine Website erstellt, bloß um dann festzustellen, dass der Konkurrenzbrowser diese völlig falsch anzeigt. Mit der zunehmenden Verbreitung von PDAs und Handhelds, die wiederum eigene Anforderungen an eine Website stellen, wird es zunehmend schwieriger, die Website für die verschiedenen Clients anzupassen. Oft findet man sich in der Lage, parallel mehrere Versionen der Website pflegen zu müssen.

Bei HTML werden die Inhalte oft nicht gesondert vom Layout getrennt, sondern sie sind stark miteinander verwoben. Genau dies macht eine Adaption für eine andere Plattform sehr schwierig.

XML will diese Lücke schließen. Die Inhalte werden in eine Struktur gefasst, und dabei völlig vom Layout getrennt. Für das Layout sind indes andere Mechanismen zuständig, die wir heute auch kurz vorstellen möchten.

## Begrifflichkeiten

Bevor wir zu ein paar Beispielen kommen, möchten wir zunächst einige Begrifflichkeiten klären. Einiges wird Ihnen schon von den vorherigen Tagen bekannt sein, einiges ist neu.

## Tags

XML arbeitet wie HTML auch auf Basis von Tags. Ein Tag wird durch eine spitze Klammer eingeleitet und ebenso abgeschlossen:

<para>

Jeder Tag muss durch einen entsprechenden »Gegenspieler« abgeschlossen werden. Dieser wird durch einen Schrägstrich eingeleitet:

</para>

Alles, was zwischen dem Start- und dem End-Tag steht, gehört zu dem Tag:

<para>All dies gehört zum Tag</para>

Innerhalb eines Tags dürfen auch andere Tags verwendet werden. Sie müssen nur darauf achten, dass sich die Tags nicht überschneiden. Folgendes ist beispielsweise korrekt:

<para>Dies ist der Inhalt des <strong>Tags</strong></para>

Folgendes ist dagegen falsch:

<para>Dies ist der <strong>Inhalt des Tags</para></strong>

Ein Tag kann auch einen oder mehrere Attribute haben. Deren Werte müssen entweder in doppelte oder in einfache Anführungszeichen eingeschlossen werden:

<para align="left" align='right'>All dies gehört zum Tag</para>

Nun gibt es auch Tags, die keinen Inhalt haben (ein Beispiel hierfür ist etwa das <br>-Tag von HTML). Auch dieses Tag muss geschlossen werden:

<br></br>

Hierfür gibt es aber auch eine Abkürzung: Der Schrägstrich kommt vor der schließenden spitzen Klammer, muss aber durch ein Leerzeichen von dem Tagnamen getrennt werden:

<br />

XML unterscheidet - im Gegensatz zu HTML - zwischen Groß- und Kleinschreibung. Folgendes ist also ebenfalls nicht korrekt:

<Para>All dies gehört zum Tag</para>

Und wenn Sie sich gefragt haben, welche Bedeutung das <para>-Tag überhaupt hat: Die Antwort lautet, das bestimmen Sie! Das X in XML steht für eXtensible, und bedeutet, dass Sie beliebig Tags definieren und festlegen können. Wie gesagt, bei XML werden Inhalt und Layout voneinander getrennt, Sie haben also bei der Tagvergabe freie Hand (sollten sich aber nichtsdestotrotz Ihre Gedanken machen).

## Entity

Sie kennen bereits von HTML her die besondere Bedeutung des kaufmännischen Unds: Hierdurch werden Sonderzeichen ausgegeben, beispielsweise *&auml;* für das kleine ä, sowie *&amp;* für das kaufmännische Und. So etwas nennt man *Entity*, und Sie können in XML auch eigene Entities erstellen.

Zum Einbau dieser Entities kommen wir später.

## Wurzelement

Das Wurzelement eines XML-Dokuments ist das oberste Element auf der Seite. Jede XML-Seite darf nur ein einziges Wurzelement haben. Bei HTML-Seiten war dieses Element `<html>`.

## Ein XML-Beispiel

XML-Dateien selbst sind wie HTML-Dateien auch reine Textdateien, können also mit den einfachsten Editoren erstellt werden. Hier ein einfaches XML-Dokument:

### Listing 21.1: autoren1.xml

```
<?xml version="1.0"?>
<autoren>
 <autor>
 <name>Dirk Louis</name>
 </autor>
</autoren>
```

Die erste Zeile ist ein Muss - hiermit wird das Dokument als XML-Dokument ausgezeichnet. Als nächstes kommt das Wurzelement, in diesem Fall `<autoren>`. Dieses enthält ein Unterelement `<autor>` mit einem weiteren Unterelement, nämlich `<name>`, welches als Inhalt den Namen eines der beiden Autoren dieses Buches enthält.

In der Praxis ist das XML-Dokument natürlich viel länger, und enthält mehrere Autoren, und womöglich auch ein paar Werke dieser Autoren:

### Listing 21.2: autoren2.xml

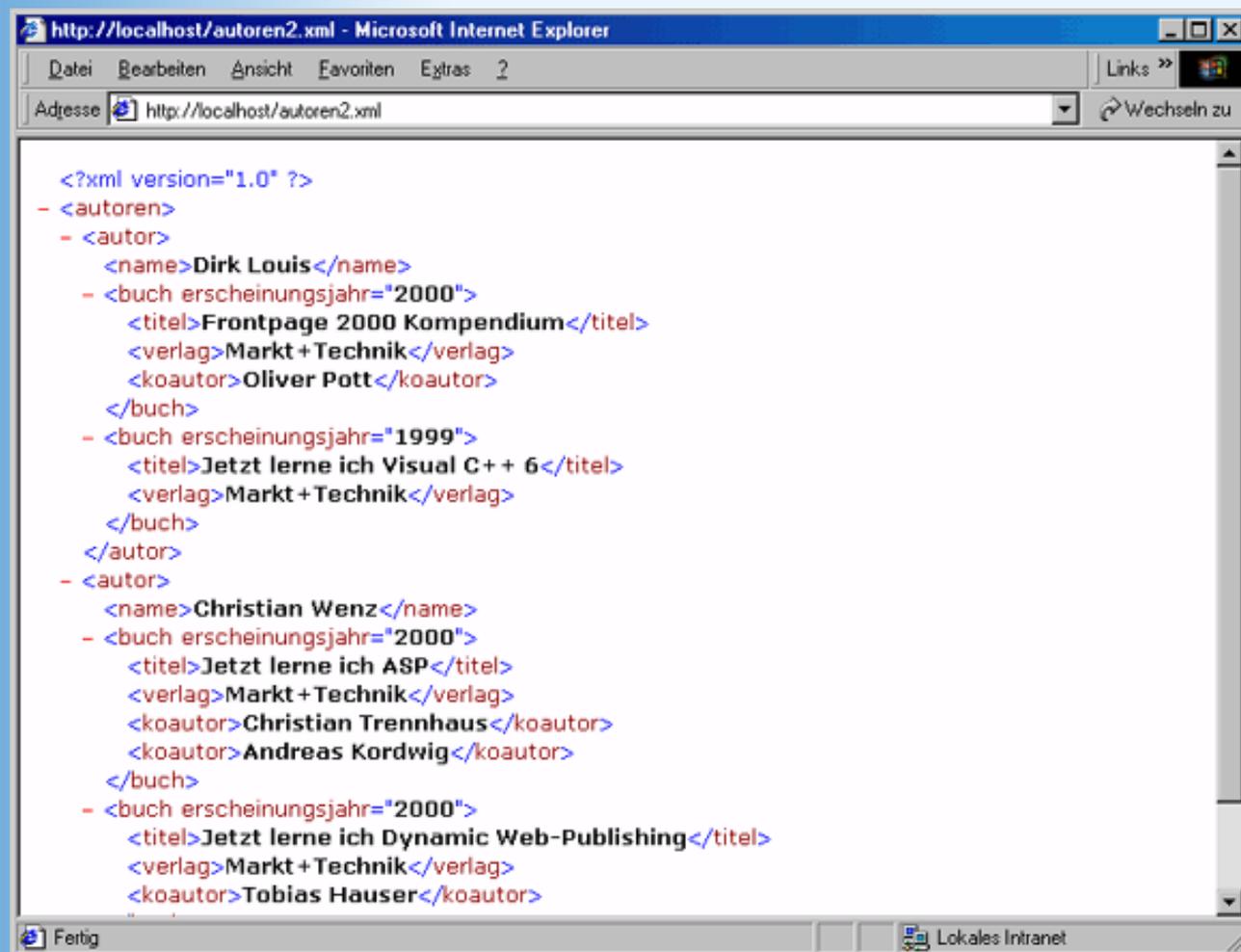
```
<?xml version="1.0"?>
<autoren>
 <autor>
 <name>Dirk Louis</name>
 <buch erscheinungsjahr="2000">
 <titel>Frontpage 2000 Kompendium</titel>
 <verlag>Markt+Technik</verlag>
 <koautor>Oliver Pott</koautor>
 </buch>
 <buch erscheinungsjahr="1999">
 <titel>Jetzt lerne ich Visual C++ 6</titel>
 <verlag>Markt+Technik</verlag>
 </buch>
 </autor>
 <autor>
 <name>Christian Wenz</name>
 <buch erscheinungsjahr="2000">
 <titel>Jetzt lerne ich ASP</titel>
 <verlag>Markt+Technik</verlag>
 <koautor>Christian Trennhaus</koautor>
 <koautor>Andreas Kordwig</koautor>
```

```

</buch>
<buch erscheinungsjahr="2000">
 <titel>Jetzt lerne ich Dynamic Web-Publishing</titel>
 <verlag>Markt+Technik</verlag>
 <koautor>Tobias Hauser</koautor>
</buch>
</autor>
</autoren>

```

Der Microsoft Internet Explorer ab Version 5 unterstützt XML; wenn Sie der Datei die Endung *.xml* geben, können Sie sie im Browser betrachten.



**Abbildung 21.2: #Die XML-Datei im Internet Explorer 5**

Nun kommt der Verlagsname Markt+Technik öfters vor (und aus naheliegenden Gründen wird ausschließlich dieser Verlagsname verwendet). Hier könnten wir auch eine Entity verwenden. Diese wird wie folgt deklariert:

**Listing 21.3: autoren3.xml (Ausschnitt)**

```

<!DOCTYPE autoren [
<!ENTITY mut "Markt+Technik">
]>

```



Beachten Sie, dass nach `<!DOCTYPE` der Name des Wurzelements kommen muss!

Nun kann sie wie folgt eingesetzt werden:

```
<verlag>&mut ; </verlag>
```

Wir drucken obiges Listing nicht noch einmal ab, wohl aber die Ausgabe des veränderten Listings im Internet Explorer. Wie Sie sehen, wurde die Entity `&mut`; wie gewünscht durch den Text »Markt+Technik« ersetzt.

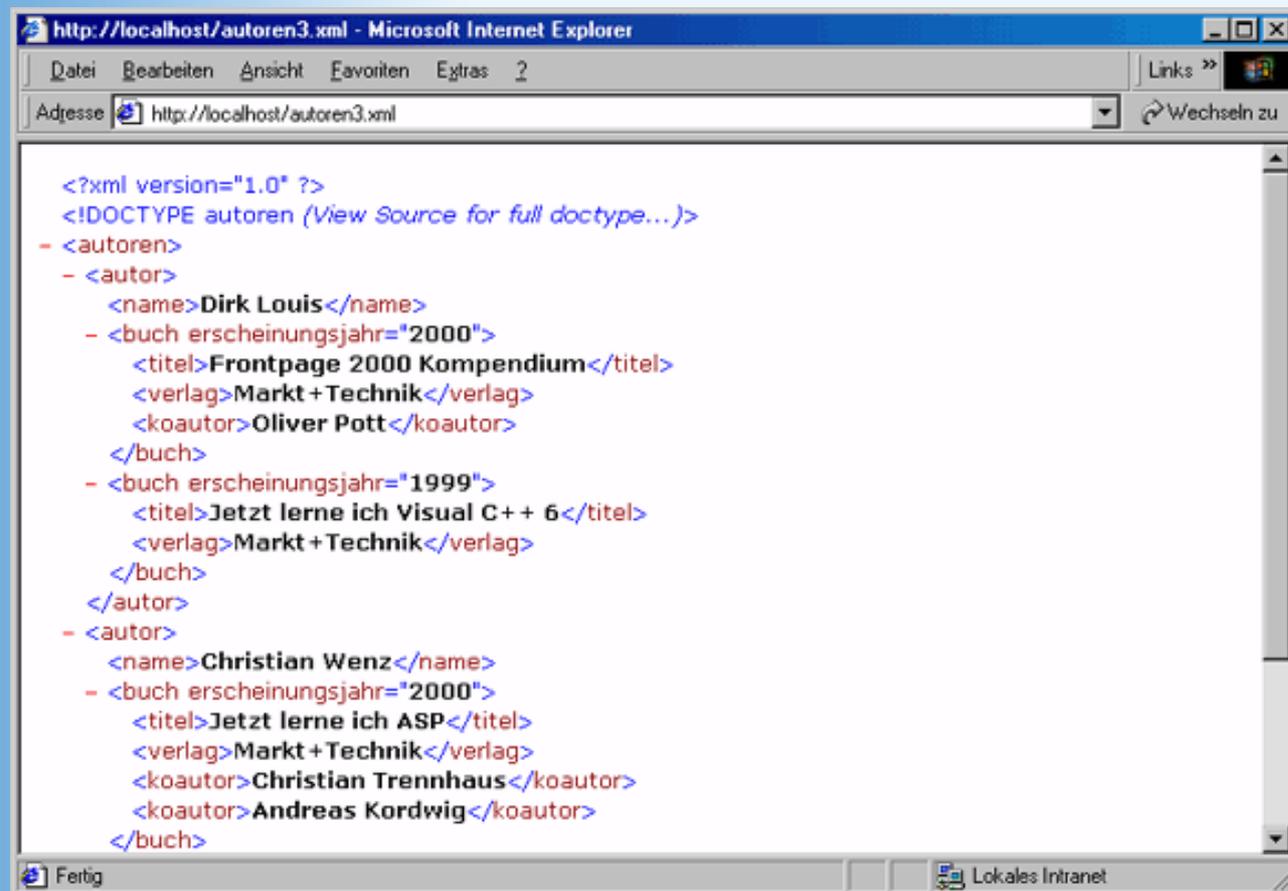


Abbildung 21.3: Entities in XML

## DTD

Zu einem XML-Dokument gehört auch noch die entsprechende DTD - *Document Type Definition*. In dieser wird angegeben, welche Struktur dem XML-Dokument zugrunde liegt, also was das Wurzelement ist, und welche Elemente unterhalb welchen anderen Elements erlaubt sind. Die DTD ist nicht notwendig, ein XML-Dokument kann auch ohne sie existieren. Im Sinne einer sauberen Strukturierung der Daten sollten Sie aber DTDs anlegen.

Eine DTD wird durch `<!DOCTYPE` eingeleitet, wie wir das bereits zuvor bei den Entities gesehen haben. Dahinter folgt der Name des Wurzelements des XML-Dokuments. Alle weiteren Angaben (beispielsweise Entities) sind in eckigen Klammern eingeschlossen.

```
<!DOCTYPE autoren [
 ...

```

## Elemente

Ein Element hat den folgenden Aufbau:

```
<!ELEMENT Name Inhalt>
```

Um das Beispiel von zuvor zu nehmen: Innerhalb von `<autoren>` gibt es das Element `<autor>`, welches die Unterelemente `<name>` und `<buch>` hat:

```
<!DOCTYPE autoren [
 <!ELEMENT autor (name, buch)>

```

Dementsprechend darf beispielsweise das Element `<koautor>` nicht direkt unterhalb von `<autor>` erscheinen. Wenn Sie einem Element gestatten möchten, dass jedes beliebige andere Element in ihm erlaubt ist, können Sie das Schlüsselwort `ANY` verwenden:

```
<!ELEMENT egal ANY>
```

Innerhalb von `<egal>` ist nun jedes Element erlaubt.

Das Gegenteil hierzu ist das Schlüsselwort `EMPTY`, welches ein leeres Element darstellt:

```
<!ELEMENT br EMPTY>
```

Nun ist es in unserem XML-Dokument aber so gewesen, dass es innerhalb von `<autoren>` mehrere Autoren geben kann, ein Autor schreibt unter Umständen auch mehrere Bücher, mit einer variablen Anzahl von Koautoren. Auch dies muss in der DTD angegeben werden. Werfen wir zunächst einen Blick auf das Element `autor`:

```
<!ELEMENT autor (name, buch)>
```

Das bedeutet: Es gibt zwei Unterelemente, `name` und `buch`. Die Reihenfolge ist hiermit festgeschrieben, und jedes dieser Unterelemente muss genau einmal vorkommen. Wenn Sie aber nun die Anzahl der Bücher variabel gestalten möchten, müssen Sie den Quantifizierer (deutsche Übersetzung des englischen *quantifier*) `+` verwenden. Dieser zeigt an: Dieses Element tritt mindestens einmal auf, aber beliebig oft. Und da man nur dann (Buch-)Autor ist, wenn man mindestens ein Buch veröffentlicht hat, ist dies genau der Quantifizierer, den wir benötigen:

```
<!ELEMENT autor (name, buch+)>
```

Wenn wir uns nicht nur auf Buchautoren beschränken möchten, sondern auch Zeitschriftenautoren berücksichtigen wollen, müssen wir noch die Alternative einführen, dass ein Autor Artikel verfasst hat. Mit dem Zeichen `|` wird diese Alternative angegeben. Das erinnert nicht ohne Absicht stark an einige der zuvor vorgestellten Programmiersprachen - der senkrechte Strich entspricht einem *oder*.

```
<!ELEMENT autor (name, (buch+ | artikel+))>
```

Der Fall, dass ein Autor sowohl Bücher als auch Artikel verfasst hat, wurde hier nicht betrachtet. Dazu benötigen wir einen anderen Quantifizierer, den wir im Folgenden vorstellen.

Werfen wir zunächst einen genaueren Blick auf das Element buch. Ein Buch hat einen Titel, einen Verlag - und eine variable Anzahl von Koautoren. Der Quantifizierer + ist also hier fehl am Platze, Sie müssen stattdessen \* verwenden, was für beliebig viele steht (insbesondere auch 0).

```
<!ELEMENT buch (titel, verlag, koautor*)>
```

Somit lässt sich auch die Definition des autor-Elements verfeinern, wenn man Zeitschriftenartikel berücksichtigen will. Folgendes ist aber falsch:

```
<!ELEMENT autor (name, buch*, artikel*)>
```

Der Grund: Wenn ein Autor kein Buch und keinen Artikel geschrieben hat, würde er dennoch auf dieses Muster passen, was wir nicht beabsichtigen. Eine funktionierende Alternative wäre Folgendes:

```
<!ELEMENT autor (name, ((buch+, artikel*) | (buch*, artikel+)))>
```

Zur Erläuterung: Entweder mindestens ein Buch und beliebig viele Artikel, oder umgekehrt.

Was ist nun (beispielsweise) mit dem Element <titel>? Auch dies muss definiert werden. Es enthält keine Unterelemente mehr, aber Text (den Buchtitel nämlich). Hierfür gibt es einen eigenen Typ, genannt CDATA. In einem CDATA-Element dürfen weitere Elemente und Auszeichnungen vorkommen. Wollen Sie dies untersagen, also insbesondere nur Text zulassen, können Sie #PCDATA verwenden (für *parseable character data*). Wir verwenden an dieser Stelle CDATA, da dies flexibler ist.

```
<!ELEMENT titel (CDATA)>
```



*Es gibt auch noch andere Typen außer CDATA und #PCDATA, beispielsweise für numerische Werte. Diese Einschränkung erweist sich in der Praxis aber als eher unpraktisch, und Strings sind nun einmal der wohl flexibelste Datentyp.*

Hier nun die vollständigen Definitionen aller Elemente unserer Autoren-XML-Datei:

```
<!DOCTYPE autoren [
<!ELEMENT autor (name, buch+)>
 <!ELEMENT name (CDATA)>
 <!ELEMENT buch (titel, verlag, koautor*)>
 <!ELEMENT titel (CDATA)>
 <!ELEMENT verlag (CDATA)>
 <!ELEMENT koautor (CDATA)>
>
```



*Natürlich wäre es möglich, koautor gegebenenfalls auch als Element vom Typ autor zu deklarieren:*

```
<!ELEMENT koautor (CDATA | koautor)>
```

Dadurch würden wir aber eine rekursive Datenstruktur erhalten, was nicht in unserem Sinne ist

## Attribute

Eines unserer Tags - und zwar <buch> - hatte auch ein Attribut, das Erscheinungsjahr. Attribute sollten in der DTD möglichst direkt nach der Elementsdefinition angegeben werden, und zwar nach folgendem Schema:

```
<!ATTLIST Element
 Attribut Typ
 Attribut Typ
 ...>
```

Für den Typ gibt es mehrere Möglichkeiten, und auch hier ist CDATA eine häufige Wahl:

```
<!ATTLIST buch
 erscheinungsjahr CDATA>
```

Wird nichts Weiteres angegeben, sind Attribute optional. Wenn Sie ein Attribut zum Muss erklären möchten, müssen Sie in der Attributliste noch #REQUIRED angeben:

```
<!ATTLIST buch
 erscheinungsjahr CDATA #REQUIRED>
```

Somit ist die DTD für unsere Autorenliste vollständig; Sie finden Sie im Folgenden noch einmal abgedruckt:

### Listing 21.4: autoren.dtd

```
<!DOCTYPE autoren [
<!ELEMENT autor (name, buch+)>
 <!ELEMENT name (CDATA)>
 <!ELEMENT buch (titel, verlag, koautor*)>
 <!ATTLIST buch
 erscheinungsjahr CDATA #REQUIRED>
 <!ELEMENT titel (CDATA)>
 <!ELEMENT verlag (CDATA)>
 <!ELEMENT koautor (CDATA)>
<!ENTITY mut "Markt+Technik">
]>
```

## Einbau

Um die DTD in Ihr XML-Dokument einzubauen, können Sie sie direkt nach der Zeile `<?xml version="1.0"?>` einsetzen. Wenn Sie der DTD jedoch mehrere Dokumente zugrundelegen möchten, ist es empfehlenswert, sie in einer eigenen Datei abzuspeichern. Als Dateiendung hat sich der Standard `.dtd` durchgesetzt (in unserem Fall beispielsweise sollten Sie die Datei `autoren.dtd` nennen). Sie können die DTD dann wie folgt in Ihre XML-Dokumente einbinden:

```
<!DOCTYPE autoren SYSTEM "autoren.dtd">
```

Natürlich können Sie auch relative und absolute Pfade angeben:

```
<!DOCTYPE autoren SYSTEM "/dtd/autoren.dtd">
```

Zu guter Letzt kann die DTD auch auf einem anderen Server liegen:

```
<!DOCTYPE autoren SYSTEM "http://ihrserver/dtd/autoren.dtd">
```



*Beachten Sie aber unbedingt Folgendes: Wenn Sie die DTD in eine externe Datei auslagern, müssen Sie das `<!DOCTYPE [ sowie ]>` entfernen - denn in der XML-Datei wird ja bereits `<!DOCTYPE` verwendet!*

## XML-Parser

Um die fertigen XML-Dateien einmal auszutesten, benötigen Sie ein entsprechendes Anzeigeprogramm. Sie können die Datei jedoch zuvor noch auf syntaktische Korrektheit prüfen.

### Datei prüfen

Ein XML-Dokument heißt *wohlgeformt*, wenn alle XML-Regeln eingehalten worden sind, also insbesondere nur ein Wurzelement, keine ineinander verschachtelten Tags, und so weiter. Die syntaktische Korrektheit eines Dokuments ist in XML besonders wichtig. Die Programmierer sind durch HTML ein wenig »verdorben«, denn die Webbrowser akzeptieren (in begrenztem Maße) auch fehlerhaftes HTML. Bei XML ist das nicht mehr der Fall, ein Dokument ist entweder wohlgeformt oder nicht. Einer der ersten Syntaxprüfer war *expat*. Sie können Version 1.x unter [www.jclark.com/xml/expat.html](http://www.jclark.com/xml/expat.html) herunterladen. Sie finden dort auch einen Link auf Version 2.0, die sich zur Zeit in Entwicklung befindet.



*Expat ist der zugrundeliegende XML-Parser im Netscape 6*

Sie finden unter der angegebenen URL den kompletten Quellcode und können Expat selbst kompilieren.

Windows-Nutzer finden im Unterverzeichnis *bin* eine bereits kompilierte und einsatzbereite Datei.

Um das Ganze zu testen, können Sie die zuvor entwickelte XML-Datei verwenden. Bauen Sie aber ein paar Fehler ein. Hier ein Beispiel. Sehen Sie die Unterschiede?

### Listing 21.5: fehlerhaft.xml

```
<?xml version="1.0">
<!DOCTYPE autoren SYSTEM "autoren.dtd">
<autoren>
 <autor>
 <name>Dirk Louis</name>
 <buch erscheinungsjahr="2000">
 <titel>Frontpage 2000 Kompendium</titel>
 <verlag>&mut;</verlag>
 <koautor>Oliver Pott</koautor>
 </buch>
 <buch erscheinungsjahr="1999">
 <titel>Jetzt lerne ich Visual C++ 6</titel>
 <verlag>&mut;</verlag>
 </buch>
 </autor>
</autoren>
<autoren>
 <autor>
 <name>Christian Wenz</name>
 <buch erscheinungsjahr="2000">
 <titel>Jetzt lerne ich ASP</titel>
 <verlag>&mut;</verlag>
 <koautor>Christian Trennhaus</koautor>
 <koautor>Andreas Kordwig</koautor>
 </buch>
 <buch erscheinungsjahr="2000">
 <titel>Jetzt lerne ich Dynamic Web-Publishing</titel>
 <verlag>&mut;</verlag>
 <koautor>Tobias Hauser</koautor>
 </buch>
 </autor>
</autoren>
```

Rufen Sie nun den Parser auf, indem Sie auf Kommandozeilenebene `xmlwf dateiname` aufrufen (in unserem Beispi etwa `xmlwf autoren.xml`). Der Parser sollte folgende Meldung ausgeben:

```
autoren.xml:1:0: unclosed token
```

Erster Fehler entdeckt: Das Fragezeichen vor der schließenden Klammer in der ersten Zeile fehlt. Fügen Sie dieses Fragezeichen hinzu und rufen Sie `xmlwf` noch einmal auf. Es kommt zu folgender Ausgabe:

```
autoren.xml:17:0: junk after document element
```

Auch diese Fehlermeldung ist leicht zu erklären: Es gibt zwei Wurzelemente, das darf nicht sein. Nachdem Sie auch diesen Fehler behoben haben (entfernen des zweiten `<autoren>`-Elements und

verschieben des zweiten <autor>-Blocks in das Innere des ersten <autoren>-Elements) gibt Expat keine Fehlermeldung mehr aus, das Dokument ist also wohlgeformt.

Expat überprüft nur XML-Dateien, nicht aber eingebundene DTDs. Wenn Sie auch DTDs überprüfen möchten, ist IBM's XML4J (XML for Java) eine gute Wahl. Sie finden diesen Parser unter [www.alphaworks.ibm.com/tech/xml4j](http://www.alphaworks.ibm.com/tech/xml4j).

## Datei anzeigen

Sowohl der Internet Explorer ab Version 5 als auch der Netscape 6 verstehen (zumindest ein wenig) XML (zum aktuellen Zeitpunkt eignet sich jedoch lediglich der Internet Explorer zum Testen). Sie können also eine XML-Datei direkt in einem dieser Browser öffnen. Sie haben das bereits in Abbildung 21.2 gesehen.

Doch diese Darstellung ist sehr spartanisch. Kommen wir also zum letzten und nicht unwichtigen Schritt - dem Layout.

## Layout mit CSS und XLS

Sie haben prinzipiell zwei Möglichkeiten, XML-Dateien mit Layout zu versehen. Entweder, Sie verwenden Style Sheets, oder Sie setzen XSL ein.

### CSS

Style Sheets haben Sie bereits in der ersten Woche kennen gelernt. Sie können eine CSS-Datei wie folgt in Ihre XML-Datei einbinden:

```
<?xml:stylesheet href="autoren.css" type="text/css"?>
```

Das folgende Style Sheet zeigt exemplarisch einige Formatierungsmöglichkeiten:

#### Listing 21.6: autoren.css

```
autor {display: block; padding-top: 5pt; font-family: verdana;}
name {display: block; font-weight: bold; font-size: 16pt;}
buch {display: block; text-indent: 15pt; font-size: 12pt;}
titel {display: block; font-weight: bold; font-size: 10pt;}
verlag {display: block; font-size: 10pt;}
koautor {display: block; font-size: 10pt; font-style: italic;}
```

Und so sieht das Ganze dann im Internet Explorer aus:

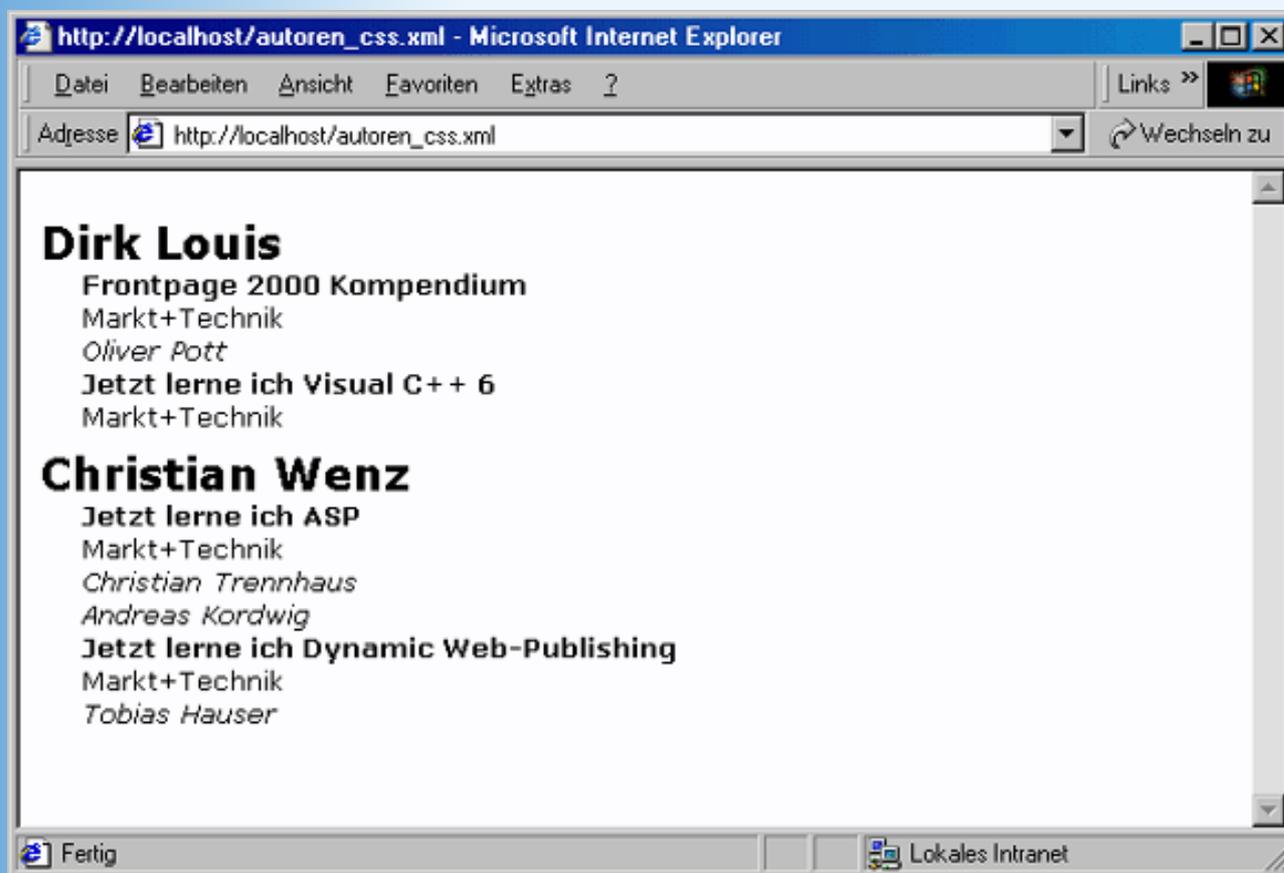


Abbildung 21.4: XML mit CSS

## XSL

XSL steht für Extensible Style Language und ist eine Sprache, mit der Layoutanweisungen für XML-Elemente angegeben werden kann. Im konkreten Fall bedeutet dies, dass die XML-Daten nach einer bestimmten Verarbeitungsvorschrift (beispielsweise) in HTML umgewandelt werden. Momentan aktuell ist XSL2. Die Sprache ist sehr umfangreich, wir können an dieser Stelle aber nur einen kleinen Ausschnitt darstellen.

Eine XSL-Datei hat den folgenden Grobaufbau:

```
<xsl:stylesheet
 xmlns:xsl="http://www.w3.org/TR/WD-xsl">
 Templates
</xsl:stylesheet>
```

Der mit *Templates* gekennzeichnete Bereich ist hier ein Platzhalter für die Formatierungsvorschriften für die einzelnen XML-Elemente. Ein einzelnes Template sieht folgendermaßen aus:

```
<xsl:template match="Muster">
 Anweisungen
</xsl:template>
```

*Muster* gibt hier an, nach welchem XML-Element oder XML-Wert gesucht werden soll. Am Einfachsten ist es, wenn Sie hier den Namen des XML-Elements angeben. Analog zur DTD-Definition können Sie auch wieder Alternativen und Ähnliches verwenden, dies soll aber nicht mehr Thema dieses Buches sein. Eine Besonderheit wollen wir nicht unerwähnt lassen: Mit / wird das Wurzelement repräsentiert.

Die *Anweisungen* werden direkt am Beispiel erläutert.

Die folgende XSL-Datei formatiert die Autorenliste. Wir drucken sie zunächst ab und analysieren Sie danach.

### Listing 21.7: autoren.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet
 xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
 <html>
 <head>
 <title>Autorenliste</title>
 </head>
 <body bgcolor="white">
 <xsl:for-each select="autoren/autor">
 <p>
 <xsl:value-of select="name"/>
 </p>
 <p>
 <xsl:apply-templates select="buch"/>
 </p>
 <hr />
 </xsl:for-each>
 </body>
 </html>
</xsl:template>
<xsl:template match="buch">
 <xsl:value-of select="titel"/><
 (<xsl:value-of select="@erscheinungsjahr"/>)

 <xsl:value-of select="verlag"/>

 <xsl:apply-templates select="koautor"/>

</xsl:template>
<xsl:template match="koautor">
 <xsl:value-of />
</xsl:template>
</xsl:stylesheet>
```

Beginnen wir mit dem ersten Block:

```
<xsl:template match="/">
 <html>
 <head>
 <title>Autorenliste</title>
 </head>
 <body bgcolor="white">
 <xsl:for-each select="autoren/autor">
 <p>
 <xsl:value-of select="name"/>
```

```

 </p>
 <p>
 <xsl:apply-templates select="buch" />
 </p>
 <hr />
 </xsl:for-each>
</body>
</html>
</xsl:template>

```

Dieser sagt aus: Suche das Wurzelement (match="/") und gib dann zunächst eine Reihe von HTML-Anweisungen aus. Mit <xsl:for-each select="autoren/autor"> wird eine For- Each-Schleife initiiert, die Sie bereits aus den Programmiersprachenkapiteln her kennen. Es werden alle <autor>-Elemente innerhalb des <autoren>-Elements durchsucht. Auch hier wird wieder eine Reihe von HTML-Code ausgegeben. Inmitten dieses Codes kommt dann folgendes Element:

```
<xsl:value-of select="name" />
```

Hierdurch wird der Wert des <name>-Elements ausgegeben; in unserem Beispiel ist das der Name des entsprechenden Autors.

Drei Zeilen weiter unten sehen Sie eine neue Anweisung:

```
<xsl:apply-templates select="buch" />
```

Hierdurch wird dem XSL-Prozessor gesagt, dass er nach dem <buch>-Element suchen (select="buch") und für alle gefundenen Elemente die Templates verarbeiten (apply- templates) soll. Werfen wir einen direkten Blick auf dieses Template:

```

<xsl:template match="buch">
 <xsl:value-of select="titel" />
 (<xsl:value-of select="@erscheinungsjahr" />)

 <xsl:value-of select="verlag" />

 <xsl:apply-templates select="koautor" />

</xsl:template>

```

Zunächst wird der Titel und der Verlag des Buches ausgegeben. Sie sehen hier auch, wie der Wert eines Attributs ausgegeben wird: Um ein Attribut von einem Element zu unterscheiden, wird dem Attributnamen (hier: erscheinungsjahr) ein Klammeraffe vorangestellt.

Als nächstes wird mit <ul> eine Aufzählungsliste gestartet, und mit <xsl:apply-templates select="koautor"/> die Verarbeitung der Templates für alle <koautor>-Elemente gestartet. Diese sehen so aus:

```

<xsl:template match="koautor">
 <xsl:value-of />
</xsl:template>

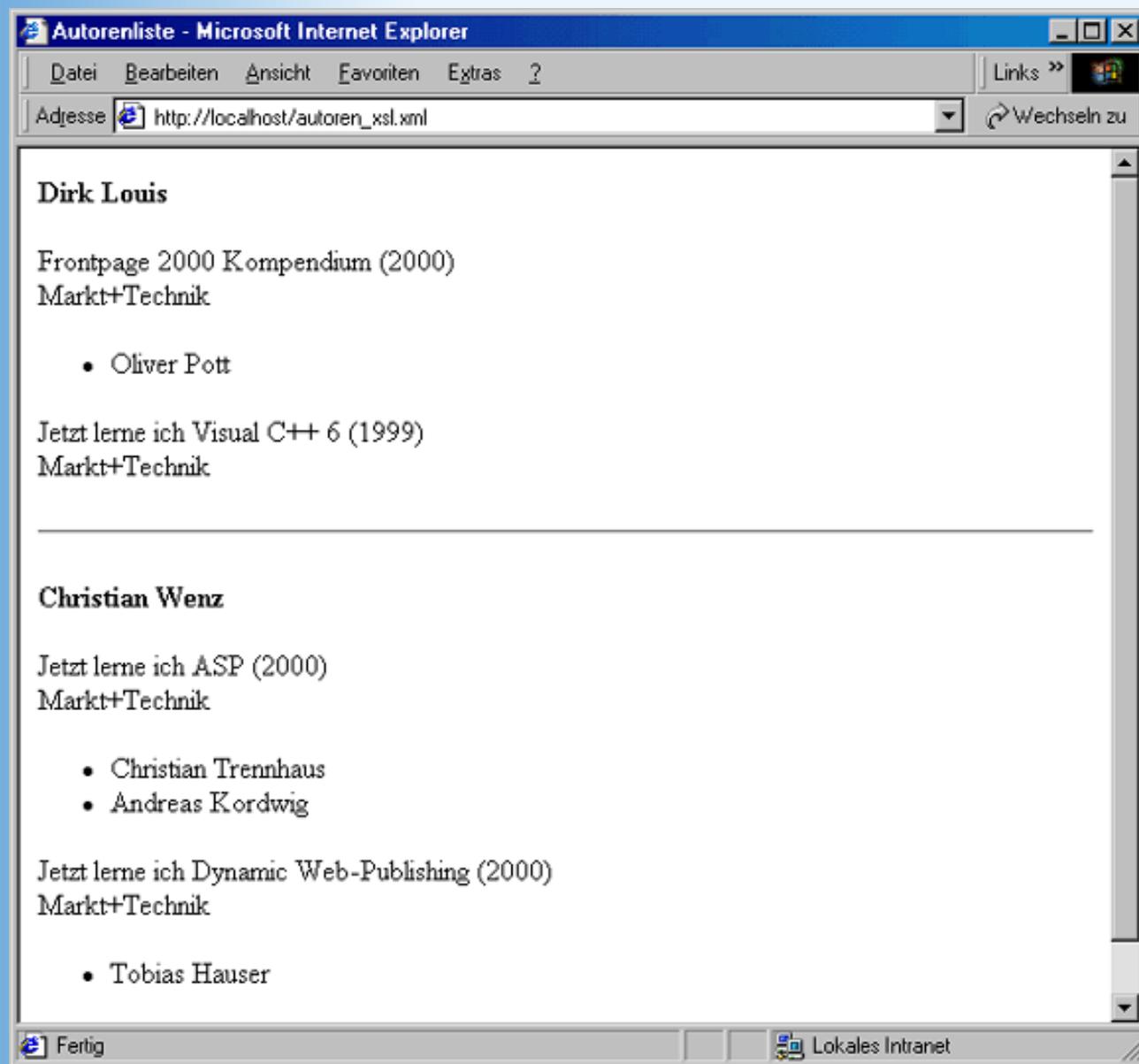
```

Der Text im Elements wird als Listenelement ausgegeben.

Und damit haben Sie Ihre erste XSL-Datei erstellt. Eingebunden wird sie in die XML- Datei wie folgt:

```
<?xml:stylesheet href="autoren.xsl" type="text/xsl"?>
```

Hierbei steht *autoren.xsl* für den Dateinamen der obigen XSL-Datei. Starten Sie Ihren Browser, und laden Sie die neue XML-Datei. In Abbildung 21.5 sehen Sie, was der Internet Explorer 5 daraus macht.



**Abbildung 21.5: XML und XSL**

Hiermit beenden wir unsere Schnelleinführung in XML. Sie haben damit einen ersten Einblick in eine Technologie erhalten, die in Zukunft eine noch größere Rolle spielen wird als jetzt schon!

## 21.2 XHTML

In Anerkennung der Bedeutung des neuen XML-Standards hat das W3C-Konsortium in den Jahren 1999/2000 seinen HTML 4-Standard auf der Basis von XML neu formuliert. Das Ergebnis heißt XHTML und lässt sich am besten als eine Art »strenges HTML« verstehen.

XHTML ist vollständig XML-kompatibel und gründet derzeit auf drei XML- Dokumenttypdeklarationen, die den Dokumenttypen aus dem HTML 4-Standard entsprechen. XHTML ist aber auch nicht weit von HTML entfernt, und unter Beachtung einiger weniger Regeln ist es möglich, Webseiten aufzusetzen, die zum

einem XML- kompatibel sind, zum anderem aber auch von traditionellen Webbrowsern ohne XML-Parser verarbeitet können.

## Warum sollte man auf XHTML umstellen?

XHTML bringt uns genau die Vorteile, die auch XML auszeichnen: es ist strikt standardisiert und es ist modular erweiterbar. Für die Zukunft ist daher zu erwarten, dass es eine Reihe von modularen Erweiterungen zu XHTML geben wird. Insbesondere die Integration neuer Technologien ist dank der modularen Erweiterbarkeit von XML/ XHTML kein Problem mehr.

XML hat gezeigt, dass es für Daten, die mit Markups gekennzeichnet sind, weit mehr Einsatzgebiete als nur die Formatierung und Darstellung in einem Browser gibt. Mit zunehmender Verbreitung von XHTML wird für XHTML eine vergleichbare Entwicklung einsetzen und es wird neben den traditionellen Browsern immer mehr XHTML-Parser geben, die XHTML-Dokumente auf die unterschiedlichsten Weisen auswerten und verarbeiten werden.

Schließlich ist XHTML wirklich nicht schwer zu erlernen. Als HTML-Autor brauchen Sie nur ein paar einfache Regeln zu beachten.

## Von HTML zu XHTML

Unter Beachtung einiger weniger Regeln ist es möglich, Webseiten aufzusetzen, die zum einem XHTML-kompatibel sind, zum anderem aber auch von traditionellen Webbrowsern ohne XML-Parser verarbeitet können.

- Beginnen Sie Ihre XHTML-Dokumente mit einer XML-Deklaration. Diese ist zwar im Grunde nur dann notwendig, wenn Sie eine andere Zeichencodierung als UTF-8 oder UTF-16 verwenden wollen, doch ist es kein schlechter Stil XHTML-Dokumente grundsätzlich mit einer XML-Deklaration zu beginnen.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- Stellen Sie dem Wurzelement <html> eine der folgenden DOCTYPE-Deklaration voraus:

```
<!DOCTYPE html
 PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "DTD/xhtml11-strict.dtd">
<!DOCTYPE html
 PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "DTD/xhtml11-transitional.dtd">
<!DOCTYPE html
 PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
 "DTD/xhtml11-frameset.dtd">
```

- Geben Sie im Wurzelement <html> den XHTML-Namensbereich an.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

Ein XML-Namensbereich ist nichts anderes als eine Sammlung von gültigen Namen (Bezeichnen) für Elementtypen und Attribute, die in einem XML-Dokument verwendet werden dürfen. Man kann dies vielleicht mit dem deutschen Alphabet vergleichen, dass den »Namensbereich« für deutsche Wörter darstellt: In einem deutschen Wort

darf man nur Buchstaben aus dem Namensbereich »deutsches Alphabet« verwenden. Da beliebige viele XML-Namensbereiche definiert werden können, müssen die einzelnen Namensbereiche eindeutig gekennzeichnet werden. Statt einer ID hat sich das W3C-Konsortium für URIs entschieden. Beachten Sie also dass der URI eines xmlns-Namensbereichs kein Link ist, den der Parser oder Browser verfolgt, sondern einfach nur ein eindeutiger Name für den zu verwendenden Namensbereich!

- Verwenden Sie das folgende Grundgerüst:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
 <title>Titel des Dokuments</title>
 </head>
 <body>
 </body>
</html>
```

Beachten Sie, dass in XHTML das <title>-Element obligatorisch ist.

- Sie dürfen HTML-Tags ineinander einbetten, aber nicht überlappen.

```
<p>Dies ist kein korrekter XHTML-Code</p>
```

- Schreiben Sie alle Tags und Attribute klein.
- Alle nicht-leeren Elemente müssen mit einem Ende-Tag abgeschlossen werden.

```
<p>Inhalt</p>
```

- Leere Elemente enden mit />.

```
<hr />
```

- Möglich ist auch <hr></hr>, doch kann dies zu Fehlinterpretationen in traditionellen Browsern führen.
- Attribut-Werte müssen immer in Anführungszeichen stehen.
- Boolesche Attribute setzt man, indem man dem Attribut seinen eigenen Namen als Wert zuweist.

```
ismap="ismap"
```

- Verwenden Sie kein name-Attribut, wo man auch ein id-Attribut definieren könnte. Insbesondere in den Tags <a>, <applet>, <form>, <frame>, <iframe>, <img> und <map> wurde das name-Attribut in XHTML als deprecated eingestuft. Weiterhin Verwendung findet es dagegen in Steuerelementen und META-Tags.
- Verwenden Sie keine Kommentare, um Skripte oder Stylesheets vor älteren Browsern zu verbergen. Der XML-Parser ist angehalten, Kommentare ganz aus dem Code zu löschen!
- XML-Parser interpretieren Zeichen wie < (&lt;) in Skripten und Stylesheets als HTML-Sonderzeichen und nicht als normale Zeichen. Man kann diesem Problem begegnen, indem man die Deklaration des eingebetteten Skripts/Stylesheets in <![CDATA[ und ]]> einfasst:

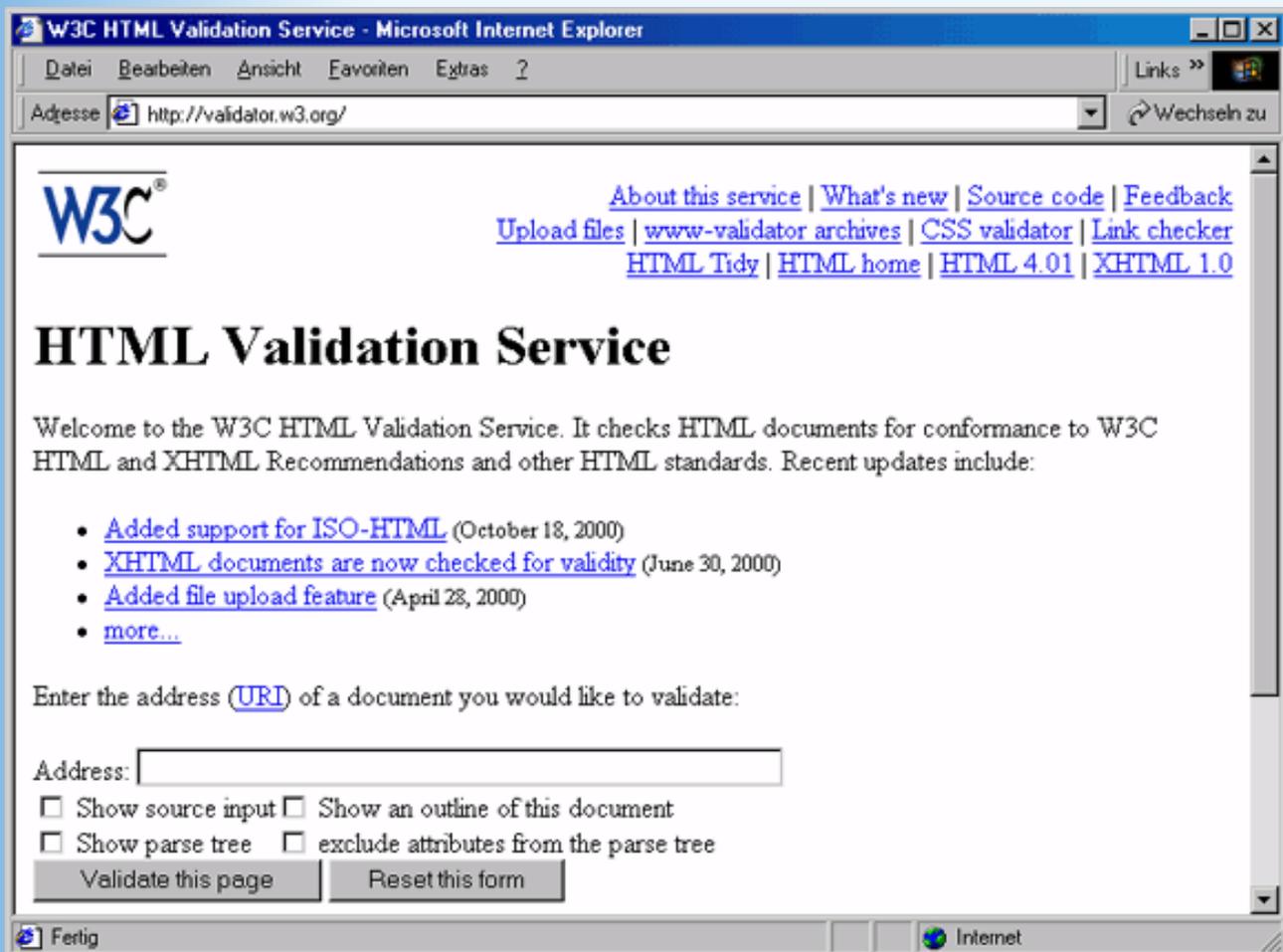
```
<head>
 <style type="text/css">
 <![CDATA[
```

```
p {color: red; font-family: sans-serif; text-align: right}
.unter {text-decoration: underline}
]]>
</style>
</head>
```

Noch besser ist es allerdings, externe Stylesheets zu verwenden, da diese auch von traditionellen Browsern verarbeitet werden können (im Gegensatz zu Stylesheets, die mit `<![CDATA[` beginnen).



Auf der Webseite <http://validator.w3.org/> können Sie Ihre HTML-Dokumente auf korrekte XHTML-Syntax hin untersuchen lassen.



**Abbildung 21.6:** Hier können Sie Webseiten online auf korrektes HTML/XHTML testen lassen

## 21.3 Zusammenfassung

Heute haben wir Ihnen XML vorgestellt. Sie haben erfahren, warum derzeit ein so großes Interesse an XML besteht und wie man als Webdesigner XML nutzen kann. Wir haben zusammen ein erstes XML-Dokument erstellt und mit Hilfe eines XML-Parsers getestet. Da XML in erster Linie nur den logischen Aufbau eines Dokuments beschreibt, stellte sich die Frage, wie man Formatierungsvorschriften für XML-

Dokumente vorsehen kann. Mit CSS und XSL haben wir Ihnen zwei Möglichkeiten zur Formatierung von XML-Daten aufgezeigt.

Zum guten Schluss sind wir noch einmal auf XHTML zu sprechen gekommen und haben eine Checkliste erstellt, nach der man weitgehendst sicherstellen kann, dass eine Webseite sowohl von traditionellen Browsern als auch von Browsern mit XML-Parsern dargestellt werden kann.

## 21.4 Fragen und Antworten

**Frage:**

**Ich finde XML sehr interessant. Sollte ich mich weiter in XML einarbeiten?**

*Antwort:*

*Auf jeden Fall. Selbst wenn es in Ihrem Tätigkeitsfeld derzeit keine sinnvolle Einsatzmöglichkeit für XML geben sollte, wird es ihr Schaden nicht sein, wenn Sie sich eingehender mit XML beschäftigen. XML ist eine Zukunftstechnologie und gewinnt derzeit ständig an Bedeutung.*

**Frage:**

**Ich arbeite als Webdesigner und kann mich nicht für XML begeistern. Muss ich mich mit XML beschäftigen?**

*Antwort:*

*Nein, es steht derzeit nicht zu erwarten, dass HTML von XML oder XHTML verdrängt oder gar ersetzt werden könnte. Wenn Sie professionell im Bereich Webdesign tätig sind, sollten Sie aber darauf gefasst sein, dass Kunden oder Ihr Chef irgendwann mit der Forderung nach XML-Anwendungen an Sie herantreten könnten.*

## 21.5 Workshop

Der Workshop enthält Quizfragen, die Ihnen helfen sollen, Ihr Wissen zu festigen, und Übungen, die Sie anregen sollen, das eben Gelernte umzusetzen und eigene Erfahrungen zu sammeln. Versuchen Sie, das Quiz und die Übungen zu beantworten und zu verstehen, bevor Sie zur Lektion des nächsten Tages übergehen.

### Quiz

1. Unterscheidet XML zwischen Groß- und Kleinschreibung?
2. Mit welchen drei Elementen beginnt ein typisches XML-Dokument?
3. Wie kann man leere HTML-Elemente so deklarieren, dass Sie sowohl von traditionellen HTML-Browsern als auch von XHTML-Parsern akzeptiert werden?

### Übungen

1. Am letzten Tag wollen wir Sie nicht noch mit Übungen quälen. Falls Sie aber trotzdem Lust haben, noch eine Übung zu machen, dann verwandeln Sie doch folgenden HTML-Code in ein korrektes XHTML-Dokument und testen Sie es unter <http://validator.w3.org/>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<head>
 <style type="text/css">
```

```
<!--
body { background-color: black; color: white;
 margin-top: 10px}
div.sonne { position: absolute; left: 350px; top: 120px}
div.erde { position: absolute; left: 335px; top: 200px }
-->
</style>
</head>
<body>
<h1>Erde und Sonne</h1>
<div class="sonne"></div>
<div class="erde"></div>
</body>
</html>
```

---

**❖ Kapitel Inhalt Index *SAMS* ❖ Top Kapitel❖**

---

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

# Tag Anhang A

## Einrichtung lokaler Webserver

Bevor Sie Ihre Webs im Internet veröffentlichen, sollten Sie sie auf einem lokalen Webserver testen. Dies gilt ganz besonders dann, wenn Sie Ihre Webseiten mit dynamischen Inhalten versehen, die auf serverseitigen Technologien beruhen - beispielsweise Formulare, die von CGI-Programmen auf dem Server ausgewertet werden sollen. Solche Webinhalte können überhaupt nur mit Hilfe eines Webserverns ausgetestet werden.

Im Folgenden finden Sie Hilfe und Tipps zur Installation, Einrichtung und Verwendung dreier weit verbreiteter Webserver, die (zumindest für Privatanwender) kostenfrei bezogen werden können:

- den Apache-Webserver für Linux (und jetzt auch für Microsoft Windows)
- den PWS beziehungsweise IIS für Microsoft Windows
- den OmniHTTPd für Microsoft Windows



*Die genaue Vorgehensweise zur Installation und Konfiguration der oben genannten Webserver hängt sowohl von der von Ihnen verwendeten Betriebssystem-Version als auch der Version der Server-Software ab. Wegen der Vielfalt der möglichen Kombinationen müssen wir uns daher auf mehr oder weniger allgemeine Hinweise beschränken und bitten um Nachsicht, wenn einzelne Angaben nicht mit Ihrer Installation übereinstimmen. Dies gilt insbesondere für Pfadangaben, die nicht nur je nach Betriebssystem- und Webserver-Version variieren, sondern teils auch bei der Installation verändert werden können.*

## A.1 Der Apache-Server

Der Apache-Server ist ein absolut vollwertiger Webserver, der sowohl für den privaten als auch den kommerziellen Gebrauch geeignet ist. Er ist kostenlos erhältlich, vielfältig konfigurierbar, unterstützt CGI, SSI und PHP und kann durch Zufügung selbst programmierter Module wie durch direkte Überarbeitung seines Quellcodes beliebig ausgebaut werden.

### Bezugsquellen

Der Apache-Server gehört mittlerweile zur Standardausstattung der verschiedenen Linux-Distributionen. Wenn Sie also beispielsweise mit RedHat- oder SuSE-Linux arbeiten, stehen die Chancen gut, dass der Server bereits auf Ihrem Rechner installiert ist und ausgeführt wird. Falls

nicht, schauen Sie zuerst einmal nach, ob Sie den Apache-Server nachinstallieren können. Für RedHat installieren Sie das betreffende RPM-Package nach, für SuSE können Sie das Apache-Modul im Zuge einer Aktualisierung des Betriebssystems installieren lassen.



*RedHat bietet auf seiner Website, [www.redhat.com](http://www.redhat.com), stets die neueste RPM-Version des Apache-Servers zum Herunterladen an.*

Ansonsten können Sie den Apache-Server von der Originalwebsite <http://www.apache.org> herunterladen.



**Abbildung A.1: Die Originalwebsite des Apache-Servers**

Über die Links **Apache Server** und **Download** gelangen Sie auf eine Webseite mit Links zum Herunterladen der Server-Software. Neben weiterführenden Links, die Sie zu den Download-Dateien für die Windows-Version (die zum Zeitpunkt der Drucklegung dieses Buch noch in einem »Experimentierstatus« war) oder den Binärversionen des Apache-Servers führen, finden Sie auf dieser Webseite mehrere Links zum Herunterladen der Quelltextsammlungen zu den neuesten Apache-Versionen. Suchen Sie sich die neueste Version aus (Achtung! Manchmal werden auch Alpha- oder Beta-Versionen zum Herunterladen angeboten) und entscheiden Sie sich für eine ZIP-Datei, die Sie entpacken können.

# Installation

Wenn Sie mit Linux arbeiten, prüfen Sie zuerst einmal, ob der Apache-Server nicht vielleicht schon installiert ist. Die Programmdatei des Apache-Servers heißt `httpd` und kann mit verschiedenen Optionen von der Kommandozeile aus aufgerufen werden. Um zu testen, ob der Apache-Server installiert ist, verwenden wir die Option `-v`:

```
> httpd -v
```

Im Erfolgsfall wird Ihnen daraufhin die Versionsnummer des Servers angezeigt:

```
Server version: Apache/1.3.12 (unix) (SuSE/Linux)
Server built: Jul 30 2000 22:47:29
```

Wenn Sie stattdessen eine Fehlermeldung erhalten, dass der Befehl nicht gefunden werden konnte, muss Sie das nicht gleich entmutigen - eventuell steht `httpd` nur nicht in Ihrem Pfad. Versuchen Sie es dann mit folgenden Aufrufen:

```
> /usr/sbin/httpd -v // bevorzugtes Installationsverzeichnis
 // für RedHat und SuSE
> /usr/locale/apache/httpd -v // für Original von www.apache.prg
```

(Im Zweifelsfall halten Sie mit dem Shell-Befehl `find / -name httpd -print` nach der `httpd`-Datei Ausschau.)

## Installation unter SuSE

Der Apache-Server gehört zum Standardumfang von SuSE. Sollte er nicht installiert sein, können Sie dies mit Hilfe von `YaST2` nachholen. Rufen Sie `Yast2` von einer Konsole aus auf, wählen Sie die Option **Pakete installieren/löschen** und klicken Sie - nachdem Sie die SuSE-CD-1 eingelegt haben - auf den Schalter **Modul starten**. Wählen Sie im erscheinenden Dialogfenster als Serie **zall** (Liste aller Pakete) aus und suchen Sie im linken Fenster nach dem **apache**-Paket. Sollte vor dem Paket kein **i** (für installiert) stehen, markieren Sie das Paket und klicken Sie auf den Schalter **Übernehmen**.

## Installation unter RedHat

Laden Sie gegebenenfalls das neueste RPM-Paket für den Apache-Server von der RedHat- Website, [www.redhat.com](http://www.redhat.com), herunter.

Installieren Sie das Paket mit Hilfe des Kommandozeilen-Tools `rpm` von der Konsole aus:

```
rpm -Uvh aktuelle_apache_version.rpm
```

## Installation der Original-Quelltextversion

Wenn Sie sich Ihre Apache-Version von der originalen Apache-Website ([http:// www.apache.org](http://www.apache.org)) heruntergeladen haben, folgen Sie den Installationshinweisen unter **Apache Server/Apache HTTP Server Documentation/Compiling and Installing** (= <http://httpd.apache.org/docs/install.html>).

Nehmen wir an, Sie hätten die Datei `apache_1.3.14.tar.gz` heruntergeladen (beispielsweise nach



Sie den Links **Apache Server/Apache HTTP Server Documentation**.



**Abbildung A.2: Online-Dokumentation zum Apache-Server**

Die wichtigste Konfigurationsdatei des Apache-Servers ist die Datei *httpd.conf*, die Sie im Konfigurationsverzeichnis des Servers finden (üblicherweise */usr/local/apache/conf*, */etc/httpd* oder */etc/httpd/conf*). Ältere Versionen des Servers verwenden zusätzlich noch die Dateien *access.conf* und *srm.conf*.

Die Log-Dateien des Webserver finden Sie üblicherweise im Unterverzeichnis *logs* Ihrer Apache-Installation (standardmäßig */usr/local/apache/logs*) oder unter */var/log/httpd*.

## Dokumentenverzeichnis

Das Dokumentenverzeichnis ist das übergeordnete Verzeichnis unter dem alle auf dem Webserver veröffentlichten Webseiten und Webs abgespeichert werden müssen. Das vom Ihrem Webserver verwendete Dokumentenverzeichnis ist in der *httpd.conf*-Datei unter dem Eintrag *DocumentRoot* festgehalten. Gibt es keinen entsprechenden Eintrag, verwendet der Server höchstwahrscheinlich */home/httpd/html*, */usr/local/httpd/htdocs* oder */usr/local/apache/htdocs* als Dokumentenverzeichnis.

## Server Side Includes

Das Apache-Server ist standardmäßig meist so konfiguriert, dass er keine Server Side Includes unterstützt.

Um die Auswertung von Server Side Includes zu aktivieren, gehen Sie wie folgt vor.

1. Um festzulegen, welche Dateien nach Server Side Includes durchsucht werden sollen, müssen Sie die Datei *httpd.conf* (*srm.conf* für ältere Versionen des Webservers) öffnen und um folgenden Eintrag erweitern:

```
AddType text/x-server-parsed-html .shtml
```

Der Server wird jetzt jede Datei mit der Extension *.shtml* nach Server Side Includes durchsuchen. Wenn Sie wollen, können Sie mittels einer zweiten Zeile eine zweite Extension (beispielsweise *.shtm*) angeben.

2. Jetzt müssen Sie noch die Server Side Includes aktivieren.

Im Falle des Apache-Servers können Sie festlegen, welche Gruppen von Server Side Includes ausgeführt werden sollen. Die wichtigsten Gruppen sind Includes (Einfügen von Webseiten, Umgebungsvariablen, statistischen Daten) und ExecCGI (Ausführung CGI-Programmen und Systembefehlen, deren Ausgaben in die Webseite eingefügt werden). Aus Sicherheitsgründen sollte man sich bei nicht lokalen Webservern auf die Includes-SSIs beschränken.

Öffnen Sie die Datei *httpd.conf* (*access.conf* für ältere Versionen des Webservers). Fügen Sie die Includes-Option in die Zeile Options ein, beispielsweise:

```
Options Includes
```

oder

```
Options IncludesNOEXEC ExecCGI
```

oder

```
Options Includes ExecCGI
```

## CGI und Perl

Für Ihren Apache-Server wurde bereits ein Verzeichnis *cgi-bin* eingerichtet (je nach Version unter */home/httpd/cgi-bin* oder */usr/local/apache/cgi-bin*). Meist ist der Apache- Server so eingerichtet, dass Sie Ihre CGI-Skripte nur noch in dieses Verzeichnis kopieren müssen.

Unter Umständen müssen Sie aber noch ein Skript-Alias einrichten, das Zugriffe auf *http://servername/cgi-bin* zu dem **cgi-bin**-Verzeichnis umleitet (das ja kein Unterverzeichnis des Dokumentenverzeichnisses ist). Loggen Sie sich dann als Root ein, laden Sie die Konfigurationsdatei *httpd.conf* und erweitern Sie diese um folgenden Eintrag:

```
ScriptAlias /cgi-bin/ /home/httpd/cgi-bin/
```

```
<Directory /home/httpd/cgi-bin>
 AllowOverride None
 Options None
</Directory>
```

Starten Sie den Server danach neu (je nach Version `/etc/rc.d/init.d/httpd restart` oder `/usr/local/apache/bin/apachectl restart`).

Wenn Sie CGI-Skripte oder -Programme in das **cgi-bin**-Verzeichnis stellen, achten Sie darauf, dass die Zugriffsrechte so gesetzt sind, dass der Webserver die Programme ausführen kann. Ändern Sie die Zugriffsrechte gegebenenfalls mit Hilfe von

```
> chmod 755 progname.pl
```

## PHP

Unter <http://www.php.net> finden Sie den Quellcode von PHP. Klicken Sie auf Downloads, und laden Sie die Quellen herunter. Der Dateiname richtet sich nach der jeweils aktuellen Version; zum Zeitpunkt der Drucklegung (Februar 2001) war das die Version 4.0.4pl1, der entsprechende Dateiname ist `php-4.0.4pl1.tar.gz`.



Abbildung A.3: Die PHP-Homepage ([www.php.net](http://www.php.net))

Sie entpacken die Datei mit dem `tar`-Befehl; wenn die von Ihnen heruntergeladene Datei einen anderen Namen als den vorher angegebenen hat, müssen Sie den Aufruf entsprechend ändern:

```
tar zxfv php-4.0.4pl1.tar.gz
```

Dabei wird das Verzeichnis *php-4.0.4pl1* angelegt, in das Sie wie folgt wechseln:

```
cd php-4.0.4pl1
```

Sie können nun wählen, welche der PHP-Optionen Sie mitkompilieren möchten. Im Online-Manual finden Sie unter [www.php.net/manual/en/install.configure.php](http://www.php.net/manual/en/install.configure.php) eine komplette Auflistung der zur Verfügung stehenden Optionen. Im Allgemeinen ist aber folgende Anweisung eine gute Wahl:

```
./configure --with-mysql --with-apache=../apache_1.3.14 --enable-track-vars
```



*Dieser Aufruf setzt voraus, dass Sie Apache 1.3.14 verwenden und diesen im Verzeichnis `../apache_1.3.14` relativ zum PHP-Verzeichnis installiert haben! Sie installieren dann automatisch den MySQL-Support und konfigurieren PHP als Apache-Modul.*

Als Nächstes rufen Sie wie gewohnt `make` und `make install` auf.

```
make
make install
```

Nun müssen Sie nur noch den Apache-Webserver auf die Unterstützung von PHP konfigurieren:

```
cd ../apache_1.3.14
./configure -activate-module=src/modules/php4/libphp4.a
make
make install
```

Nun geht es darum, die zentrale Initialisierungsdatei, die *php.ini*, einzurichten und an die richtige Stelle zu kopieren. Im PHP-Verzeichnis finden Sie eine vorgefertigte Variante:

```
cd ../php-4.0.4pl1
cp php.ini-dist /usr/local/lib/php.ini
```

Sie können nun die Datei in einem beliebigen Editor anpassen. Eventuell sollten Sie den Wert bei `extension_dir` auf Ihre lokalen Gegebenheiten hin anpassen, ansonsten sind die dort anzutreffenden Einstellungen für die meisten Anwendungen völlig in Ordnung.

Als letzten Schritt müssen Sie nun noch die Datei *httpd.conf* bzw. *srm.conf* anpassen und folgende Zeile einfügen:

```
AddType application/x-httpd-php .php
```

Sie können natürlich auch eine andere Endung wählen, oder mehrere Endungen (Achtung, nur eine Endung pro Anweisung), beispielsweise *.php4*.

Stoppen und starten Sie nun Ihren Apache Server, und die PHP-Unterstützung ist fertig eingerichtet!

Windows-Anwender des Apache haben es etwas leichter, insbesondere weil sie sich das Kompilieren und Konfigurieren weitestgehend sparen. Der Server liegt als ausführbare Datei zum Download vor, PHP gibt es als ZIP-Archiv. Zur Installation und Konfiguration von PHP selbst sei auf Abschnitt 2.2.5 verwiesen, denn unter PWS/IIS geht das ganz analog. Einziger Unterschied: Sie müssen für die Apache-Konfiguration nichts in der Registry ändern, sondern nur die folgenden drei Zeilen der *httpd.conf* bzw. *srm.conf* hinzufügen:

```
ScriptAlias /php4/ "c:/path-to-php-dir/"
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php4/php.exe"
```

Nach einem Serverneustart ist die PHP-Unterstützung funktionsbereit.



*Aktuelle Informationen und Tipps zur Installation und Konfiguration finden Sie im Online-Manual unter [www.php.net/manual/en/installation.php](http://www.php.net/manual/en/installation.php).*

## ASP

Auch unter Apache ist ASP möglich! Allerdings nicht mehr kostenfrei, weswegen wir die Installation hier nicht gesondert aufführen. Wenn Sie zu diesem Thema nähere Informationen wünschen, empfehlen wir die Website [www.chilisoft.com](http://www.chilisoft.com). Dort finden Sie ein Produkt namens ChiliASP, welches ASP-Seiten auch unter Apache ermöglicht!

## A.2 Der PWS- und der IIS-Server

Microsoft liefert seine Betriebssysteme je nach Betriebssystemversion mit dem IIS-Server oder mit dessen kleinerem Bruder dem PWS (Personal Web Server) aus. Beide Server können so konfiguriert werden, dass Sie CGI mit Perl-Skripten, ASP und PHP unterstützen.

### Bezugsquellen und Installation

Die Standard-Webserver für die Windows-Plattformen, der Personal Webserver (PWS) und der Internet Information Server (IIS) werden zumeist direkt mit dem Betriebssystem ausgeliefert. Welchen Server man nutzen kann und wie dieser zu installieren ist, hängt dabei von der Betriebssystem-Version ab: Windows 95 oder Windows 98, Windows NT 4.0 oder Windows 2000, Workstation- oder Server-Version.

An dieser Stelle präsentieren wir Ihnen eine Tabelle, der Sie entnehmen können, welche Server-Software von wo wie für welches Betriebssystem zu installieren ist.

Betriebssystem	Webserver
Windows 95	<p>Verwenden Sie den Personal Webserver.</p> <p>Windows 95 wurde ausnahmslos ohne PWS ausgeliefert. Wenn Sie den Personal Webserver nachinstallieren wollen, müssen Sie ihn von der Microsoft-Website herunterladen. Er ist Teil des Windows NT 4.0 Option Pack für Windows 95 (zuletzt gefunden unter <a href="http://www.microsoft.com/ntserver/nts/downloads/Recommended/NT4OptPk">http://www.microsoft.com/ntserver/nts/downloads/Recommended/NT4OptPk</a> oder <a href="http://www.microsoft.com/msdownload/ntoptionpack/askwiz.asp">http://www.microsoft.com/msdownload/ntoptionpack/askwiz.asp</a>).</p>
Windows 98	<p>Verwenden Sie den Personal Webserver.</p> <p>Die Server-Software wird mit dem Betriebssystem zusammen ausgeliefert und kann über <b>Start/Einstellungen/Sytemsteuerung/Software/Windows Setup</b> installiert werden. Sie benötigen dazu die Betriebssystem-CD-ROM.</p>
Windows Me	<p>Die Millenium Edition scheint auf den ersten Blick eine Variante von Windows 98 zu sein, aber der Personal Web Server existiert laut Microsoft für dieses Betriebssystem nicht. Sie müssen also auf einen anderen Server umsteigen.</p>
Windows NT 4.0 Workstation	<p>Verwenden Sie den Personal Webserver.</p> <p>Der Personal Webserver für NT Workstation ist im Windows NT Option Pack enthalten (das man von der Microsoft-Website herunterladen kann, zuletzt gefunden unter <a href="http://www.microsoft.com/ntserver/nts/downloads/Recommended/NT4OptPk">http://www.microsoft.com/ntserver/nts/downloads/Recommended/NT4OptPk</a> oder <a href="http://www.microsoft.com/msdownload/ntoptionpack/askwiz.asp">http://www.microsoft.com/msdownload/ntoptionpack/askwiz.asp</a>).</p>
Windows NT 4.0 Server	<p>Verwenden Sie den Internet Information Webserver.</p> <p>Der IIS ist Bestandteil des NT Option Packs. (beispielsweise im NT 4.0 Option Pack, zuletzt gefunden unter <a href="http://www.microsoft.com/ntserver/nts/downloads/Recommended/NT4OptPk">http://www.microsoft.com/ntserver/nts/downloads/Recommended/NT4OptPk</a> oder <a href="http://www.microsoft.com/msdownload/ntoptionpack/askwiz.asp">http://www.microsoft.com/msdownload/ntoptionpack/askwiz.asp</a>).</p>
Windows 2000 Professional	<p>Verwenden Sie den Internet Information Server.</p> <p>Dieser befindet sich auf der Betriebssystem-CD-ROM und kann über die Systemsteuerung installiert werden.</p>
Windows 2000 Server	<p>Verwenden Sie den Internet Information Server.</p> <p>Dieser ist bereits in das Betriebssystem integriert.</p>

**Tabelle A.1: Windows-Betriebssysteme und lokale Webserver**



*Der Betrieb eines lokalen Webservers erfordert, dass das TCP/IP-Protokoll installiert ist. Für Anwender von Windows 95/98 oder Windows NT 4.0 Workstation bedeutet dies, dass Sie den Rechner gegebenenfalls eigenhändig als Client-Rechner konfigurieren müssen. Rufen Sie dazu **Start/Einstellungen/Systemsteuerung/Netzwerk** auf und fügen Sie - falls nicht schon geschehen - über den Schalter **Hinzufügen** die Netzwerkkomponenten **Client für Microsoft-Netzwerke** und **TCP/IP-Protokoll** hinzu.*

## Server starten und konfigurieren

### Server starten

Der PWS und der IIS werden üblicherweise automatisch gestartet und erscheinen beim Programmstart als Symbol in der Taskleiste. Über das Kontextmenü des Symbols kann der Webserver angehalten und weiter ausgeführt werden kann.

- Unter Windows NT Workstation können Sie auch über **Einstellungen/ Systemsteuerung/Dienste** nachsehen, ob der Server läuft.
- Unter Windows 2000 Server können Sie unter (Achtung, der folgende Pfad ist etwas länger) **Start/Programme/Verwaltung/Computerverwaltung**, Unterknoten **Dienste und Anwendungen/Internet-Informationdienste**, Eintrag **Standardwebsite** nachsehen, ob der Server läuft.

### Konfiguration und Betrieb

Der Personal Web Server für Windows 95/98 kann zum Teil über den Personal Web- Manager konfiguriert werden, zum Teil muss man direkt die Systemregistrierdatenbank bearbeiten (siehe nachfolgende Abschnitte zur Konfiguration für CGI mit Perl, ASP oder PHP).

Unter Windows NT/2000 erfolgt die Konfiguration über den Internetdienst-Manager, der für NT Workstation zusammen mit den NT 4.0 Option Pack installiert wird und danach über **Start/Programme/Windows NT Option Pack/ Microsoft Personal Web Server/Internetdienst-Manager** aufgerufen werden kann. Bei Windows 2000 Server ist der Internetdienst-Manager wie auch der IIS-Server in das Betriebssystem integriert und kann über **Start/Programme/Verwaltung/Internetdienste-Manager** aufgerufen werden. Darüber hinaus kann der Webserver auch unter Windows NT/2000 über die Systemregistrierung konfiguriert werden.



*Unter Windows NT/2000 steht Ihnen eine Online-Hilfe mit Informationen zur Konfiguration und zum Betrieb des PWS/IIS zur Verfügung. Rufen Sie den Internet Explorer auf und steuern Sie die URL <http://localhost/iishelp> an.*

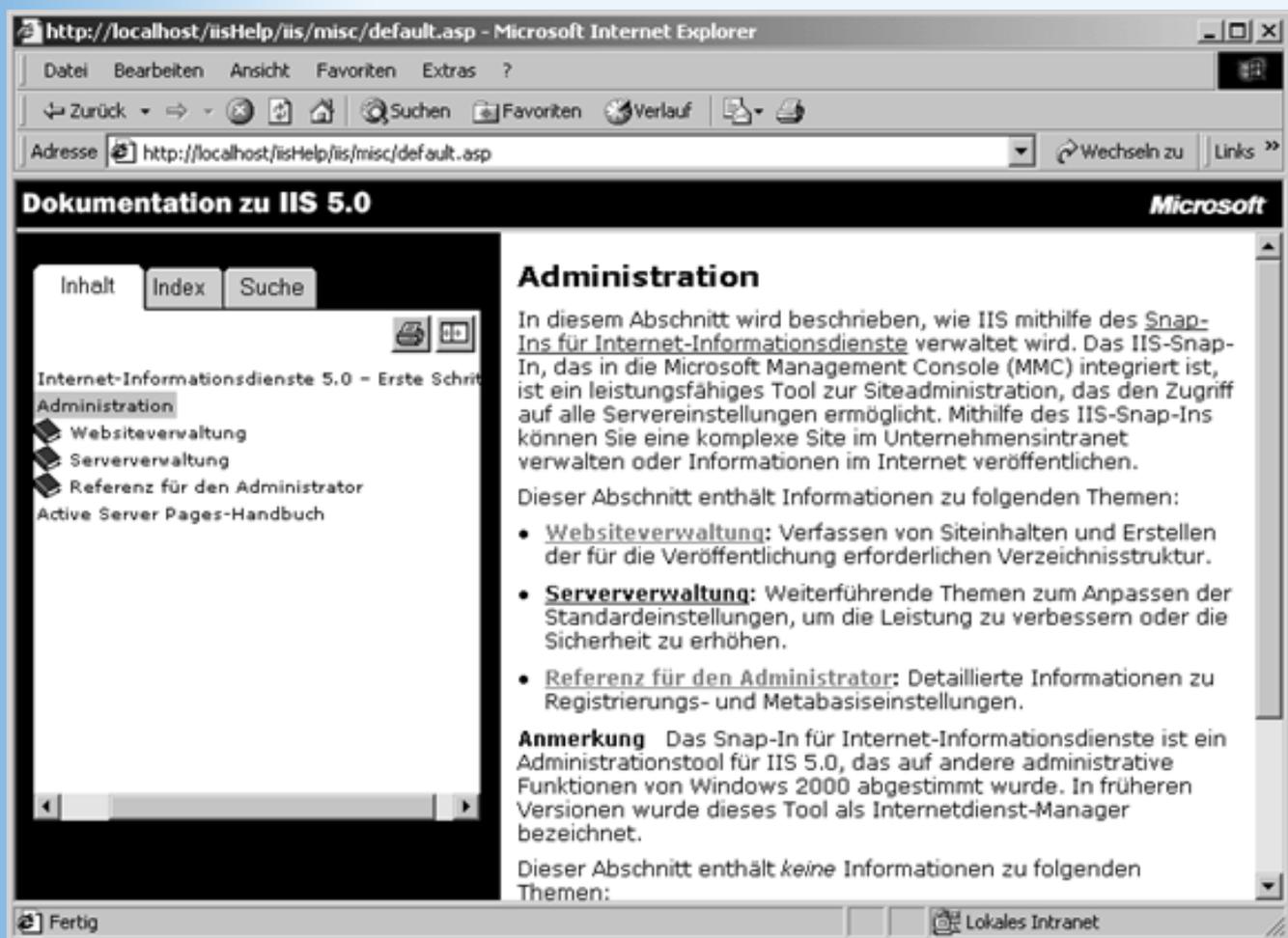


Abbildung A.4: Online-Hilfe zu PWS/IIS unter Win NT/2000

## Dokumentenverzeichnis

Das Dokumentenverzeichnis ist das übergeordnete Verzeichnis unter dem alle auf dem Webserver veröffentlichten Webseiten und Webs abgespeichert werden müssen. Das Dokumentenverzeichnis des Personal Web Servers (wie auch des IIS- Webservers) lautet standardmäßig `C:\inetpub\wwwroot`. Sie können entweder auf Ihrer Festplatte nachsehen, ob ein entsprechendes Verzeichnis angelegt wurde oder das Verwaltungsprogramm zu Ihrem Webserver aufrufen. Im Falle des Personal Web-Managers brauchen Sie dazu nur auf das Symbol des Webservers in der Taskleiste doppelzuklicken. Wenn Sie mit dem IIS- Webserver arbeiten, müssen Sie das Administrationstool für die Internet- Informationsdienste aufrufen (für Windows 2000 Server erfolgt der Aufruf über **Start/ Programme/Verwaltung/Computerverwaltung**, Unterknoten **Dienste und Anwendungen/Internet-Informationendienste**, Befehl **Eigenschaften** im Kontextmenü zu **Standardwebsite**, Registerkarte **Basisverzeichnis**).

## Server Side Includes

Der Personal Webserver ist standardmäßig so eingerichtet, dass er Server Side Includes in Webseiten mit den Extensionen `.shtml`, `.shtm` und `.stm` unterstützt.

Unter Windows NT/2000 kann man über den Internetdienst-Manager (siehe oben) eigene Extensionen festlegen. (Unter Windows 98 geht dies unseres Wissens nur durch direkte Manipulation der Systemregistrierung.)

Klicken Sie im Internetdienst-Manager mit der rechten Maustaste auf den Eintrag für die Website und

wählen Sie den Befehl **Eigenschaften** aus. Wechseln Sie in dem erscheinenden Dialogfenster zur Registerkarte **Basisverzeichnis** (oder **virtuelles Verzeichnis** oder **Verzeichnis**) und klicken Sie auf die Schaltfläche **Konfiguration**. In dem daraufhin erscheinenden Dialog ist aufgelistet, welche Dateierweiterungen mit welchen Programmen/Plug-Ins zur Bearbeitung durch den Webserver verbunden sind. Scrollen Sie das Fenster, bis Sie die Einträge für die Server Side Includes sehen, und legen Sie nach deren Beispiel neue Einträge fest oder löschen Sie bestehende Einträge.

Damit der Webserver die Dateien mit den Server Side Includes wie gewünscht auswertet, müssen Sie gegebenenfalls noch sicherstellen, dass für die Verzeichnisse, in denen die HTML-Dokumente mit den SSIs liegen, Skript- und Ausführungsberechtigung gesetzt sind.

## CGI und Perl

Wie Sie den Personal Web Server oder den IIS einrichten, hängt von der Server-Version und dem verwendeten Betriebssystem ab. Ausführlichere Informationen hierzu finden Sie in der Dokumentation zu Ihrem Server. Die Hilfedateien werden standardmäßig im virtuellen Verzeichnis *IISHelp* abgespeichert, das meist dem Windows-Unterverzeichnis *Help/IIS* entspricht. Halten Sie dort zum Beispiel nach der Datei *iicacgia.htm* Ausschau. (Sie können die Hilfe zum Server auch über Ihren Browser, <http://localhost/iishelp>, aufrufen.)

Falls Sie ActivePerl installiert haben, können Sie auch in der HTML-Dokumentation nachschlagen (*C:/Perl-Verzeichnis/html/index.html*).

## PHP

Gehen Sie zu [www.php.net](http://www.php.net) und laden Sie dort (unter »Downloads«) die Binärdistribution von PHP4 herunter. Sie haben dort die Auswahl zwischen einem kleineren und einem größeren Paket. Das kleinere Paket kommt mit einem eigenen Installationsprogramm, das PHP automatisch installiert. Wenn Sie jedoch alle Zusatzbibliotheken haben möchten, müssen Sie das größere Paket herunterladen und PHP dafür von Hand installieren.

Entpacken Sie zunächst die ZIP-Datei in ein Verzeichnis, am besten *c:\php*. Sie benötigen dazu einen Entpacker wie beispielsweise WinZip, Shareware-Version erhältlich unter [www.winzip.com](http://www.winzip.com).

Kopieren Sie nun die Datei *php.ini-dist* in Ihr Windows-Verzeichnis (beispielsweise *c:\windows* oder *c:\winnt*) und nennen Sie sie dort in *php.ini* um. Ändern Sie ggf. die Datei wie zuvor beschrieben (beispielsweise Anpassung von *extension\_dir* auf *c:\php\extensions*).

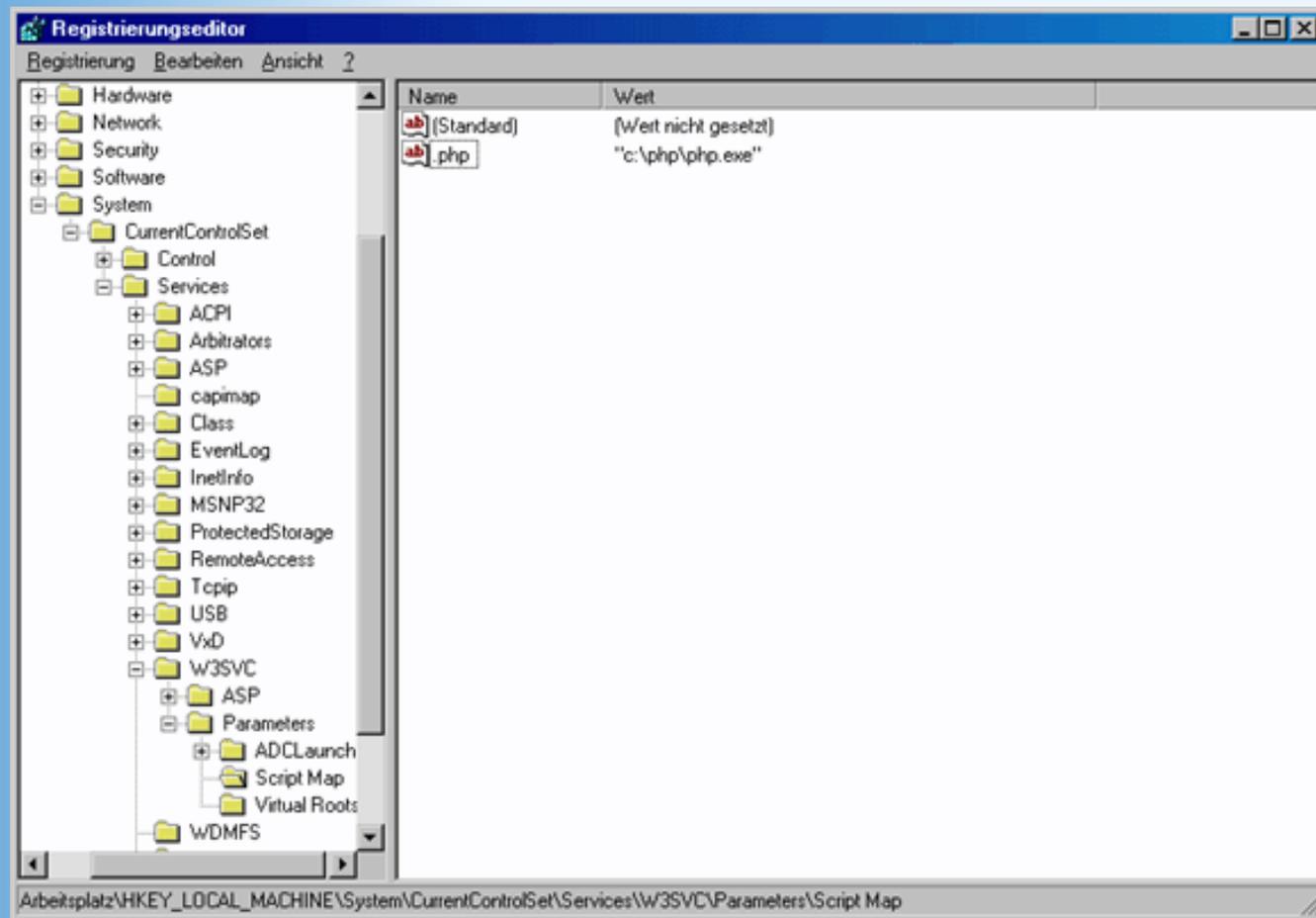
Als nächstes kopieren Sie die Datei *php4ts.dll* in das Windows-Systemverzeichnis (also etwa *c:\windows\system* oder *c:\winnt\system32*).

Im letzten Schritt müssen Sie nur noch dem Webserver mitteilen, was er mit Dateien mit der Endung *.php* (oder *.php4*) anfangen soll. Beim Personal Web Server müssen Sie dazu die Registry bearbeiten. Da diese die zentralen Konfigurationseinstellungen für Ihr System beinhaltet, sollten Sie an dieser Stelle sehr exakt vorgehen!

Wählen Sie den Befehl **Start/ Ausführen** und geben Sie **regedit** ein. Wechseln Sie nun zum Registry-Schlüssel **HKEY\_LOCAL\_MACHINE/System/ControlSet/**

**Services/W3SVC/Parameters/ScriptMap** und erstellen Sie einen neuen Schlüssel mit Namen *.php* (bzw. der von Ihnen präferierten Dateiendung) und dem Wert *c:\php\php.exe* (bzw. den

entsprechenden Pfad zu Ihrer PHP-Installation).



**Abbildung A.5: Die Einstellungen in der Registry für die PHP-Unterstützung des PWS**

Benutzer des Internet Information Server starten die Management Console (zumeist unter **Start/ Programme/ Windows NT 4.0 Option Pack/ Microsoft Internet Information Server/ Internetdienst-Manager**. Rufen Sie mit einem Klick auf die rechte Maustaste das Kontextmenü der Website aus und wählen dort den Eintrag **Eigenschaften**. Klicken Sie im Register **Basisverzeichnis** auf die Schaltfläche **Konfiguration**, und erstellen Sie mit **Hinzufügen** eine neue Zuordnung. Als Erweiterung geben Sie *.php* an (oder *.php4* oder wie gewünscht), als Programm *c:\php\php.exe*.

Sie müssen nun den Webserver komplett neu starten. Unter Windows geht das - so traurig das hier klingen mag - am besten mit einem kompletten Neustart des Systems.

## ASP

Die ASP-Unterstützung muss beim PWS/IIS nicht extra eingerichtet werden; der Webserver ist automatisch auf die Unterstützung von ASP-Seiten vorbereitet. Auch die entsprechenden Bibliotheken zur Unterstützung von VBScript und JScript sind installiert.

## A.3 Der OmniHTTPd-Server

Der OmniHTTPd-Server ist ein absolut vollwertiger Webserver mit Unterstützung für Perl, SSI und PHP. Gegenüber dem PWS/IIS besticht er durch seine einfache Installation und Konfigurierbarkeit.

## Bezugsquellen

Den OmniHttpd-Webserver von Omnicron kann man über die Website [http:// www.omnicron.ab.ca](http://www.omnicron.ab.ca) beziehen.

Neben der Vollversion, die man käuflich erwerben kann, gibt es immer auch eine aktuelle kostenlose Version, die man für den privaten Gebrauch herunterladen kann. Diese besitzt zwar ein Verfallsdatum, kann aber vor dieser Frist durch eine neuere, kostenfreie Version ersetzt werden.



Abbildung A.6: Website zum Herunterladen des OmniHTTPd-Webserver

## Installation

Nachdem Sie die exe-Datei des OmniHTTPd-Webserver auf Ihre Festplatte heruntergeladen haben, brauchen Sie diese nur doppelt anzuklicken, um den Installationsvorgang zu starten. Folgen Sie den Anweisungen in den Dialogfeldern und richten Sie den Server so ein, dass er bei dem Booten automatisch geladen wird.

Läuft der Server, erscheint in der Windows-Taskleiste ein entsprechendes Symbol für den Server. Läuft der Server nicht, kann er über seine Programmgruppe (Aufruf über **Start/ Programme**) gestartet werden.



*Sie können keine zwei Webserver gleichzeitig ausführen, wenn diese den gleichen Port*

*(für Webseiten üblicherweise 80) überwachen. Wenn Sie also den OmniHTTPd-Server verwenden wollen, achten Sie darauf, dass Sie nicht bereits zuvor schon einmal den PWS oder IIS installiert haben und dieser ebenfalls automatisch beim Booten gestartet wird.*

## Server starten und konfigurieren

### Server starten

Wenn der Server nicht bereits automatisch beim Booten des Betriebssystems gestartet wird, können Sie ihn jederzeit über die bei der Installation eingerichtete Omnicron- Programmgruppe manuell starten.

### Konfiguration

Um den Webserver anzuhalten, weiter auszuführen oder zu schließen, brauchen Sie nur mit der rechten Maustaste auf das Symbol des Webserver in der Taskleiste zu klicken und den entsprechenden Befehl aufzurufen.

Zur Konfiguration des Webserver rufen Sie das Dialogfenster **Configuration** auf. Klicken Sie dazu mit der rechten Maustaste auf der Symbol des Webserver in der Taskleiste und rufen Sie den Befehl **Properties** auf oder wählen Sie den Befehl **OmniHTTPdAdministration** in der Omnicron- Programmgruppe auf.



*Standardmäßig wird der OmniHTTPd-Webserver bereits automatisch für die Verwendung von CGI mit Perl, SSI und PHP eingerichtet. Auf der Standardstartseite unter <http://localhost/DEFAULT.HTM> finden Sie Links zum Testen dieser Technologien sowie zur allgemeinen Konfiguration und zum Betrieb des Servers.*



Abbildung A.7: Online-Dokumentation zum Omnicron-Webserver

## Dokumentenverzeichnis

Das Dokumentenverzeichnis des OmniHTTPd-Webserver ist das Unterverzeichnis `httpDocs`. Sofern Sie OmniHTTPd also in das Verzeichnis `C:/httpd` installiert haben, lautet das Dokumentenverzeichnis `C:/httpd/HtDocs`.

## Server Side Includes

Der OmniHTTPd-Webserver ist standardmäßig so eingerichtet, dass er Server Side Includes in Webseiten mit der Extension `.shtml` unterstützt.

Um eine andere Dateierweiterung festzulegen, rufen Sie über die Programmgruppe des OmniHTTPd-Webserver das Administrationsprogramm auf. Klicken Sie auf den Schalter **Web Server Global Settings** und wechseln Sie in dem erscheinenden Dialog zur Registerkarte **MIME**. Scrollen Sie in dem Listenfeld zu dem Eintrag:

```
wwwserver/html-ssi .shtml
```

Markieren Sie den Eintrag, geben Sie im Feld **Actual** eine andere Dateierweiterung ein und klicken Sie auf einen der Schalter zum Hinzufügen (**Add**) oder Ersetzen (**Replace**).

Um sicherzustellen, dass der Webserver die Dateien mit den Server Side Includes wie gewünscht auswertet, wechseln Sie zur Registerkarte **Advanced** und vergewissern Sie sich, dass die Option

Process Server Side Includes (SSI) aktiviert ist.

## CGI und Perl

Klicken Sie in der Windows-Taskleiste mit der rechten Maustaste auf das Symbol des OmniHTTPd-Servers und rufen Sie den Befehl **Properties** auf. In älteren Versionen wechseln Sie daraufhin zum Register »**Advanced**«, aktivieren dort die Option **Enable Perl CGI Support**, tragen im Eingabefeld **Perl CGI Command Line** den Pfad zu Ihrem Perl-Interpreter ein (beispielsweise `C:\perl\bin\perl.exe`) und starten danach den Server neu.

In der aktuellen Version (2.0.7) klicken Sie im **Configuration**-Dialog auf den Schalter **Web Server Global Settings**, wechseln zur Registerkarte **External**, markieren nacheinander die Einträge für `.pl` und `.plx` und passen jeweils den Pfad so an, dass er zu dem auf Ihrem Rechner installierten Perl-Interpreter führt.

## PHP

Laden Sie die PHP-Executables für Windows wie im Abschnitt 2.2.5 beschrieben herunter. Die Variante mit dem automatischen Installationsprogramm unterstützt leider nicht OmniHTTPd, Sie müssen also das größere Archiv (mit allen Erweiterungsbibliotheken) wählen. Zunächst müssen Sie wieder die Datei `php.ini-dist` in Ihr Windows-Verzeichnis kopieren und in `php.ini` umbenennen sowie die Datei `php4ts.dll` ins Windows-System- Verzeichnis kopieren.

Klicken Sie nun mit der rechten Maustaste auf das OmniHTTPd-Icon in der Windows- Taskleiste und wählen Sie den Befehl **Properties** an. Unter **Web Server Global Settings** können Sie im Register **External** analog zur Perl-Installation die entsprechende Dateierweiterung (`.php`) mit dem entsprechenden Programm (`c:\php\php.exe`) assoziieren. Erstellen Sie zusätzlich noch einen weiteren Eintrag, und zwar mit dem Wert `wwwserver/stdcgi` unter *virtual* sowie `.php` unter *actual*.

## ASP

Leider ist für OmniHTTPd keine ASP-Unterstützung vorgesehen, und uns ist auch kein Produkt bekannt, das OmniHTTPd um ASP-Fähigkeiten erweitert. Sie sind hier mit den Servern von Microsoft besser beraten.

Je nach Apache-Version kann das Verzeichnis auch `/usr/local/apache/cgi-bin/` lauten. Sie können aber auch ein eigenes Verzeichnis für Ihre CGI-Programme festlegen.

# Tag Anhang B

## Sonderzeichen

Es ist möglich, eine Reihe von Sonderzeichen in der HTML-Seite zu verwenden (beispielsweise die deutschen Umlaute). Alle Sonderzeichen beginnen mit einem &, gefolgt von einer mehr oder weniger einfach zu merkenden Buchstabenfolge und einem Semikolon.

Zeichen	Beschreibung	Name in HTML
	Erzwungenes Leerzeichen	&nbsp;
"	Anführungszeichen	&quot;
&	Kaufmännisches Und	&amp;
<	kleiner	&lt;
>	größer	&gt;
!	umgekehrtes Ausrufezeichen	&iexcl;
¢	Cent-Zeichen	&cent;
£	Pfund-Zeichen	&pound;
¤	Währungszeichen	&curren;
¥	Yen-Zeichen	&yen;
	durchbrochener Strich (*)	&brvbar;
§	Paragraph-Zeichen	&sect;
¨	Pünktchen oben	&uml;
©	Copyrightzeichen	&copy;
ª	Ordinal-Zeichen weiblich	&ordf;
«	angewinkelte Anf.zeichen links	&laquo;
	Verneinungszeichen	&not;

-	kurzer Trennstrich	&shy;
®	Registriermarke-Zeichen	&reg;
ˉ	Überstrich	&macr;
°	Gradzeichen	&deg;
±	Plusminuszeichen	&plusmn;
²	Hoch-2-Zeichen	&sup2;
³	Hoch-3-Zeichen (*)	&sup3;
´	Acute-Zeichen	&acute;
μ	Mikrozeichen	&micro;
¶	Absatzzeichen	&para;
·	Mittelpunkt	&middot;
¸	Häkchen unten	&cedil;
¹	Hoch-1-Zeichen (*)	&sup1;
º	Ordinal-Zeichen männlich	&ordm;
»	angewinkelte Anf.zeichen rechts	&raquo;
¼	ein Viertel (*)	&frac14;
½	ein Halb (*)	&frac12;
¾	drei Viertel (*)	&frac34;
¿	umgekehrtes Fragezeichen	&iquest;
À	A mit Accent grave	&Agrave;
Á	A mit Accent acute	&Aacute;
Â	A mit Circumflex	&Acirc;
Ã	A mit Tilde	&Atilde;
Ä	A Umlaut	&Auml;
Å	A mit Ring	&Aring;
Æ	A mit legiertem E	&AElig;
Ç	C mit Häkchen	&Ccedil;
È	E mit Accent grave	&Egrave;
É	E mit Accent acute	&Eacute;
Ê	E mit Circumflex	&Ecirc;

Ë	E Umlaut	&Euml;
Ì	I mit Accent grave	&lgrave;
Í	I mit Accent acute	&iacute;
Î	I mit Circumflex	&lcirc;
Ï	I Umlaut	&iuml;
Ð	Eth (isländisch) (*)	&ETH;
Ñ	N mit Tilde	&Ntilde;
Ò	O mit Accent grave	&Ograve;
Ó	O mit Accent acute	&Oacute;
Ô	O mit Circumflex	&Ocirc;
Õ	O mit Tilde	&Otilde;
Ö	O Umlaut	&Ouml;
×	Malzeichen (*)	&times;
Ø	O mit Schrägstrich	&Oslash;
Ù	U mit Accent grave	&Ugrave;
Ú	U mit Accent acute	&Uacute;
Û	U mit Circumflex	&Ucirc;
Ü	U Umlaut	&Uuml;
Ý	Y mit Accent acute (*)	&Yacute;
Þ	THORN (isländisch) (*)	&THORN;
ß	scharfes S	&szlig;
à	a mit Accent grave	&agrave;
á	a mit Accent acute	&aacute;
â	a mit Circumflex	&acirc;
ã	a mit Tilde	&atilde;
ä	a Umlaut	&auml;
å	a mit Ring	&aring;
æ	a mit legiertem e	&aelig;
ç	c mit Häkchen	&ccedil;
è	e mit Accent grave	&egrave;

é	e mit Accent acute	&eacute;
ê	e mit Circumflex	&ecirc;
ë	e Umlaut	&euml;
ì	i mit Accent grave	&igrave;
í	i mit Accent acute	&iacute;
î	i mit Circumflex	&icirc;
ï	i Umlaut	&iuml;
ð	eth (isländisch) (*)	&eth;
ñ	n mit Tilde	&ntilde;
ò	o mit Accent grave	&ograve;
ó	o mit Accent acute	&oacute;
ô	o mit Circumflex	&ocirc;
õ	o mit Tilde	&otilde;
ö	o Umlaut	&ouml;
÷	Divisionszeichen	&divide;
ø	o mit Schrägstrich	&oslash;
ù	u mit Accent grave	&ugrave;
ú	u mit Accent acute	&uacute;
û	u mit Circumflex	&ucirc;
ü	u Umlaut	&uuml;
ý	y mit Accent acute (*)	&yacute;
þ	thorn (isländisch) (*)	&thorn;
ÿ	y Umlaut	&yuml;

**Tabelle B.1: Verschiedene HTML-Sonderzeichen**

Eine vollständige Liste finden Sie in der HTML-Spezifikation (<http://www.w3.org>) unter dem Stichwort »character entity references«.

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

# Tag Anhang C

## HTML-Elemente

Die folgende Tabelle enthält einer Liste der standardisierten HTML-Elemente. Zu jedem Element ist angegeben, ob es ohne End-Tag ist, ob es ein leeres Element ist, ob es veraltet ist und ob es nur in der Übergangs (Ü)- oder der Frame (F)-DTD definiert ist.

Name	End-Tag	Leer	Depr.	DTD	Beschreibung
<a>					Anker
<abbr>					Kürzel
<acronym>					Akronym
<address>					Information über Autor
<applet>			X	Ü	Java-Applet
<area />	X	X			Feld zu clientseitiger ImageMap
<b>					fette Schrift
<base />	X	X			Basis-URL
<basefont />	X	X	X	Ü	Standardschriftgröße
<bdo>					übergehe BiDi
<big>					große Schrift
<blockquote>					größeres Zitat
<body>					Body-Abschnitt
 	X	X			Zeilenumbruch
<button>					Schaltfläche
<caption>					Tabellenüberschrift
<center>			X	Ü	Kürzel für <div align="center">

<cite>					Zitat
<code>					Listing
<col />	X	X			Tabellenspalte
<colgroup>					Tabellenspaltengruppe
<dd>					Definition eines Begriffs
<del>					gelöschter Text
<dfn>					Definition (Textauszeichnung)
<dir>			X	Ü	Verzeichnisliste
<div>					Container-Element
<dl>					Definitionsliste
<dt>					zu definierender Begriff
<em>					Auszeichnung
<fieldset>					Gruppe von Formularfeldern
<font>			X	Ü	Schrift
<form>					Formular
<frame>	X	X		F	Frame-Fenster
<frameset>				F	Aufteilung in Frames
<h1> bis <h6>					Überschriften
<head>					Header-Abschnitt
<hr />	X	X			Horizontale Linie
<html>					Wurzelelement
<i>					kursive Schrift
<iframe>				L	Inline-Frame
<img />	X	X			Bild
<input />	X	X			Steuerelement
<ins>					eingefügter Text
<isindex />	X	X	X	Ü	Einzeilige Eingabe
<kbd>					Eingabe (Textauszeichnung)
<label>					Beschriftung zu Formularfeld
<legend>					Feldbeschreibung

<li>					Listenelement
<link>	X	X			Link
<map>					clientseitige ImageMap
<menu>			X	Ü	Menüliste
<meta />	X	X			Metainformation
<noframes>				F	Alternative zu Frames
<noscript>					Alternative zu Skript
<object>					eingebettetes Objekt
<ol>					Nummerierung
<optgroup>					Optionsgruppen für <select>-Listen
<option />	X				Optionsfeld
<p>	X				Absatz
<param />	X	X			Parameter (für Objekte und Applets)
<pre>					vorformatierter Text
<q>					Zitat
<s>			X	Ü	durchstreichen
<samp>					Beispiel (Textauszeichnung)
<script>					Skript
<select>					Auswahlliste
<small>					kleine Schrift
<span>					Textpassage
<strike>			X	Ü	durchstreichen
<strong>					Hervorhebung
<style>					Stylesheet
<sub>					niedriger stellen
<sup>					höher stellen
<table>					Tabelle
<tbody>					Tabellenkörper
<td>					Tabellenzelle
<textarea>					mehrzeiliges Textfeld

<tfoot>				Tabellenfuss
<th>				Überschriftszelle in Tabelle
<thead>				Tabellenüberschrift
<title>				Titel
<tr>				Tabellenzeile
<tt>				Schreibmaschinenschrift
<u>		X	Ü	unterstrichen
<ul>				Aufzählung
<var>				variable (Textauszeichnung)

**Tabelle C.1: HTML-Elemente**

# Tag Anhang D

Lösungen

## D.1 Tag 1: Von der Idee zum eigenen Web

### Antworten zum Quiz

1. In der Fremde wird man anfangs immer erst nach dem Äußeren beurteilt. Es lohnt sich daher, in das eigene Äußere zu investieren, man muss nur aufpassen, dass man sich nicht übernimmt oder über das Ziel hinausschießt. Als Webautor präsentieren Sie sich mit Ihrer Website. Investieren Sie also entsprechend Ihrer Mittel Zeit und Geld, aber hüten Sie sich davor, Ihre Webseiten mit dynamischen Inhalten und Spielereien zu überfrachten.
2. Die Umsetzung der HTML-Tags in geeignete Textformatierungen ist weitgehend den Browsern überlassen. Auch werden nicht alle HTML-Elemente von allen Browsern gleichermaßen gut und buchstabengetreu unterstützt. Es ist also durchaus möglich, dass ein und dieselbe Webseite in einem Browser umwerfend und in dem nächsten dilettantisch aussieht. Testen Sie das Aussehen Ihrer Webseiten daher möglichst in mehreren Browsern, zumindest aber in den Standardbrowsern Internet Explorer, Netscape Navigator und Netscape 6.

### Antworten zu den Übungen

1. Siehe Kapitel 1.4.1 und 1.4.2.
2. Siehe Kapitel 1.4.3.
3. Siehe Kapitel 1.5.

## D.2 Tag 2: Am Anfang war... HTML

### Antworten zum Quiz

1. Ein leeres Tag ist ein Tag, das nur aus einem Start-Tag besteht und keinen Text einschließt (beispielsweise `<hr />` oder ``). Ein nicht-leeres Tag ohne Inhalt ist ein Tag, das üblicherweise einen Text einschließt, beispielsweise `<p>Dies ist der Inhalt</p>`, in diesem besonderen Fall aber leer ist.
2. Zeilenumbrüche werden auf verschiedene Weisen erzeugt
  - Durch das End-Tag eines Block-Tags (beispielsweise `<h1>`, `<p>` oder `<div>` )
  - Durch das `<br />`-Tag
  - Durch das Drücken der (Return)-Taste in einem `<pre>`-Block.
3. Diese Frage ist nicht zu beantworten, da die Formatierung von Struktur-Elementen wie `<h4>` vom jeweiligen Webbrowser abhängt.
4. Bilder werden üblicherweise mit dem `<img>`-Tag eingefügt.

```

```

Hintergrundbilder werden über die Stileigenschaft `background-image` eingefügt.

```
<body style="background-image: url('kiefer.gif');
background-repeat: no-repeat;
```

5. ImageMaps sind Bilder, in denen mehrere Teilbereiche mit Hyperlinks verbunden sind. Sie werden meist clientseitig durch eine Tabelle (<map>) von <area>-Definitionen erzeugt.
6. Inline-Stile werden mit Hilfe des style-Attributs in das Start-Tag des zu formatierenden HTML-Tags eingefügt.
7. Folgende Stileigenschaften waren gesucht

- background-color (Hintergrundfarbe)
- color (Vordergrundfarbe)
- text-indent (Erste Zeile einziehen) oder margin-left (Block einziehen)
- font-family (Schriftart)
- border-width (Rahmenbreite)

8. Um zwei Textabsätze und ein dazwischen gelegenes Bild zusammen einzurücken, fasst man die Textabsätze und das Bild in ein <div>-Tag ein und rückt den gesamten <div>-Block ein.

```
<div style="margin-left: 5em">
 <p>Erster Absatz</p>

 <p>Zweiter Absatz</p>
</div>
```

## Antworten zu den Übungen

1. Der HTML-Code für die angegebene Webseite könnte beispielsweise folgendermaßen aussehen.

### Listing 4.1: webseite.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>HTML und XHTML</title>
</head>
<body>
<h1>Webseiten mit HTML und XHTML</h1>
<h3>Inhaltsverzeichnis</h3>

 HTML
 XHTML

<hr />
<h2>1. HTML</h2>
<p>HTML (Hypertext Markup Language) ist eine Auszeichnungssprache zur
Erstellung von elektronischen Dokumenten. Zur Auszeichnung und Strukturierung
des Textes definiert der HTML-Standard eine Reihe von sogenannten Tags.</p>
<h2>2. XHTML</h2>
<p>XHTML ist der aktuelle Versuch des W3C-Konsortiums eine Brücke
zwischen HTML und XML zu schlagen. XHTML-Dokumente sind einerseits den HTML-
Dokumenten sehr ähnlich und können von allen gängigen Browsern korrekt
angezeigt werden, andererseits sind XHTML-Dokumenten XML-konform.</p>
<p>Unter http://www.w3.org finden Sie weitere Informationen zu HTML
und XHTML.</p>
</body>
```







Im Grunde sollte es ausreichen, die Stileigenschaft `margin` des `<body>`-Tags auf 0 zu setzen, um sicherzustellen, dass die Webseite direkt am Rahmen des Browserfensters beginnt. Aus Rücksicht auf den Netscape 4- und ältere IE-Browser setzen wir aber noch die Attribute `marginwidth`, `marginheight`, `leftmargin` und `topmargin`.

Damit die Bilder nahtlos verbunden werden, muss man die `<body>`-Attribute `border`, `cellpadding` und `cellspacing` auf 0 setzen. Breite und Höhe der Tabellenzellen müssen den Größen der Bilder angepasst werden (oder kleiner eingestellt werden). Für angrenzende Zellen müssen wir ebenfalls die Breite (und gegebenenfalls die Höhe) angeben, damit der Browser die Zellen mit den Bildern nicht auf Kosten der angrenzenden Zellen ausdehnt. Schließlich muss man aufpassen, dass in den Zellen (zwischen den `<td>`-Tags) kein weiterer Inhalt steht, insbesondere auch keine Leerzeichen oder Zeilenumbrüche! (In obigem Code haben wir den Zeilenumbruch daher innerhalb des `<img>`-Tags gesetzt.) [1](#)



*Als DOCTYPE haben wir in diesem Falle Transitional ohne URL-Angabe gewählt. Nur so war es uns möglich, die Bilder auch im neuen Netscape-6-Browser nahtlos aneinander zu fügen.*

3. Viele aktuelle Browser haben Probleme mit den `<colgroup>`-Definitionen. Der erste Schritt ist daher, die Formatierungen aus den `<colgroup>`-Definitionen in die Tabellenzellen zu übertragen.

Da der Navigator 4 die Stileigenschaft `padding` nicht für Tabellenzellen unterstützt (und wir das `<table>`-Attribut `cellpadding` nicht verwenden wollen), bedienen wir uns zum Abrücken der Ländernamen vom rechten Zellenrand eines Blinden GIFs (*spacer.gif*). Dabei handelt es sich um ein 1-Pixel großes Bild in der Hintergrundfarbe, das mit Hilfe des Attributs `width` auf die gewünschte Breite gestreckt wird (siehe auch Kapitel 5.4).

Schließlich fassen wir die Tabelle in `<div>`-Tags ein und rücken statt der Tabelle den ganzen `<div>`-Block ein, weil der Navigator 4 die Stileigenschaft `margin` nicht für Tabellen unterstützt.

#### Listing 4.4: Auszug aus `bananenproduktion.html`

```
<div style="margin-left: 30px">
<table border="1" width="550" rules="groups" >
 <colgroup width="120" ></colgroup>
 <colgroup span="2" ></colgroup>
<thead>
 <tr>
 <td rowspan="2"> </td>
 <th colspan="2">Bananenproduktion in Mio t</th>
 </tr>
 <tr>
 <th>1990</th>
 <th>1999</th>
 </tr>
</thead>
<tbody>
 <tr>
 <th width="130" align="right">Indien</th>
 <td width="210" align="center">6,734 </td>
 <td width="210" align="center">10,200</td>
```

```

</tr>
<tr>
 <th width="130" align="right">Brasilien</th>
 <td width="210" align="center">5,506</td>
 <td width="210" align="center">5,592</td>
</tr>
<tr>
 <th width="130" align="right">Ecuador</th>
 <td width="210" align="center">3,055</td>
 <td width="210" align="center">4,600</td>
</tr>
</tbody>
</table>
</div>

```

4. Zuerst erhält jede Gedichtüberschrift in der Datei `hauptseite.html` ihren eigenen, benannten Anker:

```

<h2>DER PANTHER</h2>
...
<h2>DER HERBSTTAG</h2>
...
<h2>Im SAAL</h2>
...

```

Dann richten wir in der Datei `inhaltsverzeichnis.html` die Hyperlinks auf die Gedichte ein (und vergessen auch nicht, den rechten Frame »hauptframe« als Zielframe für die Hyperlinks einzurichten.

#### Listing 4.5: `inhaltsverzeichnis.html`

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Inhaltsverzeichnis</title>
</head>
<body>
<dl>
 <dt>Rilke</dt>
 <dd><a href="hauptseite.html#panther"
 target="hauptframe">Der Panther</dd>
 <dd><a href="hauptseite.html#herbsttag"
 target="hauptframe">Herbsttag</dd>
 <dd><a href="hauptseite.html#saal"
 target="hauptframe">Im Saal</dd>
</dl>
</body>
</html>

```

Hinweis: Man könnte auch den Header-Abschnitt um ein `<base>`-Tag erweitern und dort den Zielframe für die Hyperlinks der Seite angeben.

5. Ein mögliches Formular für die Umfrage könnte wie folgt aussehen:

## Listing 4.6: umfrageformular.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Umfrage</title>
</head>
<body>
<h3>Was halten Sie von der Einführung des Internet-freien Sonntags?</h3>
<form action="/cgi-bin/umfrage.pl" method="get">
 <table border="0" cellspacing="0" cellpadding="10">
 <colgroup span=2">
 <col width="130">
 <col width="150">
 </colgroup>
 <tr>
 <td align="right"><input type="radio" name="R1" value="1" /></td>
 <td align="left" valign="top" width="230">
 Ich bin absolut dafür</td>
 </tr>
 <tr>
 <td align="right"><input type="radio" name="R1" value="2" /></td>
 <td align="left" valign="top" width="230">
 Ich bin strikt dagegen</td>
 </tr>
 <tr>
 <td align="right"><input type="radio" name="R1" value="3" /></td>
 <td align="left" valign="top" width="230">
 Weiss nicht</td>
 </tr>
 <tr>
 <td> </td>
 <td><input type="submit" value="Abschicken" />
 <input type="reset" value="Zurücksetzen" /></td>
 </tr>
 </table>
</form>
</body>
</html>
```

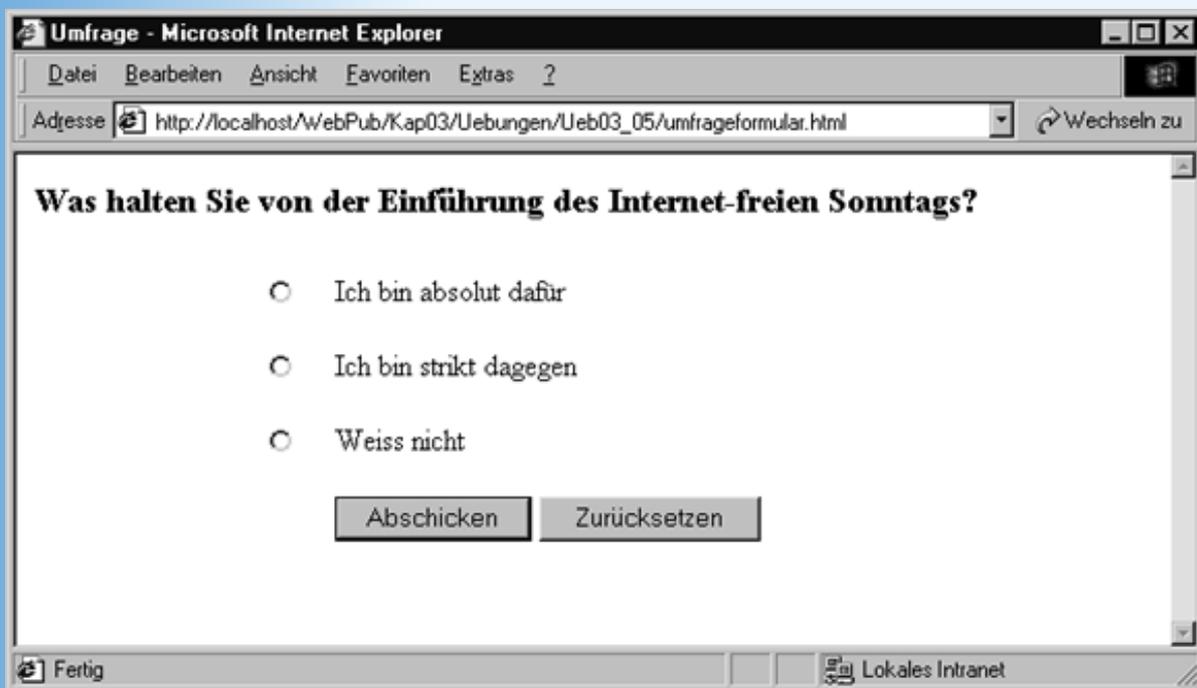


Abbildung D.2: Das Umfrageformular

## D.4 Tag 4: Formatieren mit Stylesheets

### Antworten zum Quiz

1. Eine Möglichkeit ist, den Stylesheet-Typ im <style>-Tag anzugeben.

```
<style type="text/css">
```

Die zweite Möglichkeit besteht darin, ein entsprechendes META-Tag zu definieren:

```
<meta http-equiv="content-style-type" content="text/css">
```

Diese Vorgehensweise hat zudem den Vorteil, dass sie auch für Inline-Stile gültig ist.

2. Für Webseiten ist die Stylesheet-Typangabe zwar nicht wirklich erforderlich, weil wir stets mit CSS-Stylesheets arbeiten und die Browser standardmäßig auf CSS-Stile eingerichtet sind. Es gehört jedoch zum guten Ton, die verwendete Stylesheet-Sprache anzugeben, und man sollte sich nicht unbedingt darauf verlassen, dass die Browser standardmäßig von CSS-Stylesheets ausgehen. Schließlich ist es möglich, dass die Webseite noch von anderen Anzeigegeräte verarbeitet wird (Braille-Geräte, etc.) und diese könnten auch anderen Stylesheet-Spezifikationen unterstützen.
3. Die Eigenschaft padding erzeugt einen Freiraum zwischen dem Inhalt eines Elements und seinem (sichtbaren oder nicht-sichtbaren) Rahmen. Ist eine Hintergrundfarbe für das Element definiert, füllt diese auch den padding-Raum aus. Die Stileigenschaft margin erzeugt einen Freiraum (Rand) um ein HTML-Element, der außerhalb des (sichtbaren oder nicht-sichtbaren) Rahmens liegt. Der Rand gehört nicht zum Hintergrund des Elements.
4. Die bevorzugte Stileigenschaft zum Unterstreichen von Text ist text-decoration mit dem Wert underline, beispielsweise

Ein `<span style="text-decoration: underline">unterstrichenes</span>` Wort.

Eine Alternative, die mehr Gestaltungsmöglichkeiten bietet, ist die Einblendung des unteren Rahmens:

```
<p>Ein <span style="border-width: 0 0 2px 0; border-style: solid; border-color:
```

```
red">unterstrichenes Wort.</p>
```

5. Den Inhalt von Tabellenzellen rückt man von den Zellenrändern am einfachsten über das table-Attribut cellpadding ein. Man kann aber auch die Stileigenschaft padding verwenden.
6. Vererbung bedeutet, dass bestimmte Stileigenschaften von HTML-Elementen auf eingebettete (untergeordnete) HTML-Elemente vererbt werden. So erbt ein <span>- Element beispielsweise Schriftart, Text- und Hintergrundfarbe des umliegenden Blockelements (<p>, <li>, etc.)
7. Die höchste Priorität hatte li.meineinzug, da ein class-Stil immer spezifischer ist als jede Kombination aus HTML-Selektoren.

## Antworten zu den Übungen

1. Als Beispiel-Design habe ich folgendes Stylesheets definiert, die in der Datei *meindesign.css* abgespeichert sind:

### Listing 4.7: meindesign.css - Als externes Stylesheet definiertes Standard-Design

```
body { margin-left: 10em; margin-right: 10em; text-align: justify;
 background-color: maroon; color: white;
 font-family: Times, sans-serif}
h1, h2, h3 { font-family: Western, fantasy }
ul { list-style-image: url(listsymbol.gif)}
li { padding-left: 2em}

a:link { color: lime}
a:visited { color: fuchsia}
a:active { color: white}
```

2. Zur Demonstration habe ich folgende Hauptseite aufgesetzt:

### Listing 4.8: ueb2.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Design</title>
</head>
<body>
<h1>Demo-Seite</h1>
<p>Dies ist ein Absatz mit einigen Hyperlinks. Folgen Sie doch einmal den
Hyperlinks, um zu sehen, wie besuchte und noch nicht besuchte Links dargestellt
werden. Die Darstellung aktivierter Links sehen Sie, wenn Sie einen Hyperlink
klicken und die Maustaste gedrückt halten.<p>

Erster Hyperlink
Zweiter Hyperlink
Dritter Hyperlink

</body>
</html>
```

Ohne Zuweisung des Stylesheets sieht diese Seite wie in Abbildung 2.3 aus.

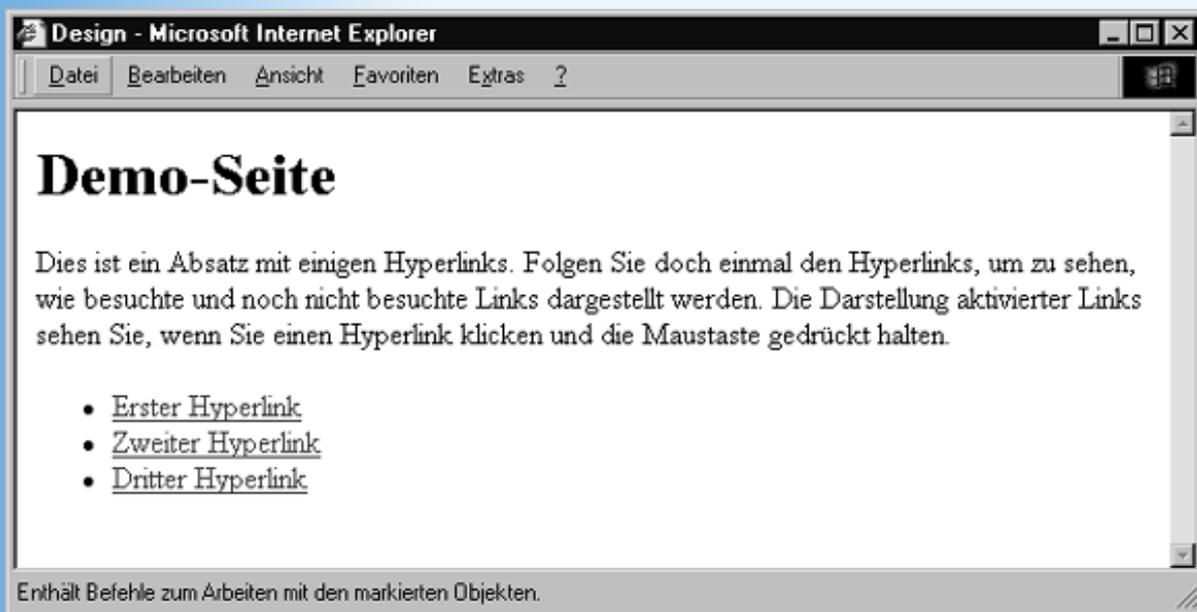


Abbildung D.3: Die Webseite ohne zugewiesenes Stylesheet

Weist man dieser Webseite per `<link>`-Eintrag im Header das Stylesheet *meindesign.css* zu,

```
<link rel=StyleSheet type="text/css" href="meindesign.css">
```

sieht die Webseite danach wie in Abbildung 2.4 aus. Das gleiche Stylesheet weist man den anderen Seiten des Webs zu (im Beispiel die Seiten *blank1.html*, *blank2.html* und *blank3.html*, auf die die Hyperlinks verweisen).



Abbildung D.4: Die »designte« Webeite.

## D.5 Tag 5: Positionieren mit Stylesheets

## Antworten zum Quiz

1. Zur absoluten Positionierung eines Elements setzen Sie die Stileigenschaft `position` auf `absolute` und positionieren das Element über die Eigenschaften `left` und `top` (oder auch `bottom` und `right`).

```

```

2. Relativ positionierte Elemente werden - nachdem der Browser die Seite auf traditionelle Weise aufgebaut hat - verschoben und hinterlassen an ihrer ursprünglichen Position einen Leerraum
3. Auf keinen Fall. Ob man besser mit Rändern, Tabellen oder absoluter Positionierung arbeitet, muss man von Fall zu Fall entscheiden.
4. Statt des deprecated `align`-Attribute für `<img>`-Bilder sollten Sie die Stileigenschaften `vertical-align` und `float` verwenden. (Hinweis: Aus Rücksicht auf ältere Browser empfiehlt es sich aber, trotzdem zusätzlich auch das `align`-Attribut zu setzen.)

## Antworten zu den Übungen

1. Für diese Übung müssen Sie die Bilder der Sonne und der Erde durch absolute Positionierung überlagern: `position: absolute; left: 350px; top: 120px`. Um sicherzustellen, dass die Erde über der Sonne liegt, können Sie die Erde entweder nach der Sonne positionieren (wie in nachfolgendem Listing) oder die `z-index`-Eigenschaft der Erde auf einen höheren Wert setzen als den `z-index` der Sonne.

Mehr Probleme als die Positionierung dürfte vielleicht der Hintergrund des Erde-Bilds machen. In der Datei `erde.gif` ist der Hintergrund nicht transparent. Folglich sieht man vor der Sonne den schwarzen Hintergrund des Erde-GIFs. Um den Hintergrund transparent zu machen, müssen Sie die Bilddatei in ein geeignetes Grafikprogramm laden (beispielsweise Paint Shop Pro), den Index der Hintergrundfarbe ermitteln (in Paint Shop Pro wird dieser in der Color-Werkzeugleiste angezeigt, wenn Sie das Dropper-Werkzeug (Pipette) über den Hintergrund bewegen), und dann das Bild mit dem Index der Hintergrundfarbe als transparente Farbe abspeichern.

### Listing 4.9: sonne.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Erde und Sonne</title>
 <style type="text/css">
 <!--
 body { background-color: black; color: white;
 margin-top: 10px}
 div.sonne { position: absolute; left: 350px; top: 120px}
 div.erde { position: absolute; left: 335px; top: 200px }
 -->
 </style>
</head>
<body>
<h1>Erde und Sonne</h1>
<div class="sonne"></div>
<div class="erde"></div>
</body>
</html>
```



**Abbildung D.5: Überlagerte Bilder mit transparentem Hintergrund**

2. Ein passender HTML-Code könnte wie folgt aussehen.

**Listing 4.10: img.html**

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Bilder ausrichten</title>
</head>
<body>
<h1>Das veraltete align-Attribut für Bilder</h1>
<div style="margin-left: 4.5cm">
 <p>Absatz mit Bild im Text (align="bottom")</p>
 <p>Absatz mit Bild im Text (align="middle")</p>
 <p>Absatz mit Bild im Text (align="top")</p>
</div>
<table border="0">
<tr>
 <th align="left"></th>
 <th align="right"></th>
</tr>
<tr>
 <td><p>Absatz mit left-align Bild im Text</p><p>Nachfolgender Text. Nachfolgender Text.
 Nachfolgender Text. Nachfolgender Text. Nachfolgender Text.
 Nachfolgender Text. Nachfolgender Text. Nachfolgender Text. Nachfolgender Text.
 Nachfolgender Text.</p>
 </td>
</tr>

```

```

</td>
<td><p>Absatz mit right-align Bild im Text</p><p style="clear: right">Nachfolgender Text.
Nachfolgender Text. Nachfolgender Text. Nachfolgender Text. Nachfolgender Text.
Nachfolgender Text. Nachfolgender Text. Nachfolgender Text. Nachfolgender Text.
Nachfolgender Text. </p>
</td>
</tr>
</table>
</body>
</html>

```

## D.6 Tag 6: Fortgeschrittenes und dynamisches HTML

### Antworten zum Quiz

1. Abspielbare Sounddateien kann man entweder über das `<embed>`-Tag

```

<embed src="MeinSong.wav" autostart="false" hidden="false"
loop="false">

```

oder durch Hyperlinks anbieten:

```

Mein Song

```

2. Quickinfos sollten aus zwei Gründen keine wichtigen Informationen enthalten:

- Quickinfos werden nicht von allen Browsern angezeigt
- Webbesucher rechnen normalerweise nicht damit, wichtige Informationen über Quickinfos zu erhalten

3. Die vollständigen URLs lauten:

- *http://www.meinServer.de/Verz /demo1.html*
- *http://www.meinServer.de/Verz/bilder/bild.gif*
- *http://www.meinServer.de/cgi-bin/prog.pl*

4. Ein Webcrawler oder Suchroboter ist ein Programm, das sich von Link zu Link durch die Webseiten des Internets hangelt und die gefundenen Webseiten gemäß ihren Schlüsselwörtern katalogisiert und in die Datenbank seiner Suchmaschine einträgt.

5. Zur Unterstützung von Suchrobotern sollte man zumindest die beiden folgenden META-Tags aufsetzen:

```

<meta name="keywords" content="Gedichte">
<meta name="description" content="Gedichtsammlung mit eigenen Werken und
Gedichten ausgewählter Dichter">

```

### Antworten zu den Übungen

1. Besorgen Sie sich zuerst eine kleine Auswahl an Sounddateien. Am einfachsten ist es, erst einmal die eigene Festplatte nach Dateien mit der Extension `.wav`, `rm` oder `.ram` zu durchsuchen.

Dann setzen Sie die Webseite mit den Links zu den Sounddateien auf:

```

Mein Song
Crossroads
...

```

Wenn Sie die fertige Webseite in Ihren Browser geladen haben, klicken Sie auf die Links, um die Sounddateien abzuspielen.



*Kann eine Sounddatei nicht abgespielt werden, fehlt Ihnen vermutlich das zugehörige Abspielprogramm. Für die meisten gängigen Soundformate gibt es kostenlose Abspielprogramme im Internet. Einige Browser führen Sie zudem automatisch zu einer passenden Downloadseite, wenn Sie versuchen, eine Datei abzuspielen, für die kein Abspielprogramm (oder -Plugin) installiert ist.*

2. Eine wie in der Aufgabe beschriebene Eingangsseite könnte beispielsweise wie folgt aussehen.

#### Listing 4.11: eingangsseite.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Hallo</title>
 <!-- Suchmaschinen unterstützen -->
 <meta name="language" content="de">
 <meta lang="de" name="keywords" content="Florfliegen, Goldaugen,
 Blattlausfliegen,
 Chrysopidae,
 Biologie, Anatomie">
 <meta name="description" content="Ausführungen zur Anatomie und
 Biologie der Florfliege (Familie Chrysopidae)">
 <meta name="revisit-after" content="30 days">
 <meta name="robots" content="all">
 <link rel="start" type="text/html" href="frame_seite.html"
 title="Ausgewählte Gedichte">
 <style type="text/css">
 <!--
 body { background-color: maroon; color: white;
 margin-top: 50px}
 h1 { font-family: 'Book Antiqua', Arial, sans-serif ;
 font-size: 24pt;
 text-align: center}
 div { float: right }
 em { visibility: hidden; color: maroon }
 -->
 </style>
</head>
<body>
<h1>Willkommen

auf

meiner

Florfliegen-Website</h1>
<p> </p>
<div></div>
<p>Hier finden Sie ausführliche Informationen über die Biologie und
Anatomie der Florfliegen (Familie Chrysopidae, auch Goldaugen oder
```

Blattlausfliegen genannt) </em><p>

</body>

</html>



Die Einrichtung unsichtbarer HTML-Elemente wurde bereits in Kapitel 5.5 beschrieben. Wir bedienen uns hier zweier Techniken: der Stileigenschaft `visibility` und der Angleichung von Vordergrund- und Hintergrundfarbe.



Abbildung D.6: Eingangseite mit unsichtbarem Text zur Suchroboterunterstützung

## D.7 Tag 7: Ein gelungener Webauftritt

Zu diesem Tag gab es keine Fragen oder Übungen.

## D.8 Tag 8: Wozu JavaScript?

### Antworten zum Quiz

1. Von serverseitiger Programmierung spricht man, wenn ein Programm auf dem Webserver ausgeführt wird und das Ergebnis über das Netz an den Browser auf dem Client-Rechner geschickt wird. Von clientseitiger Programmierung spricht man, wenn der Programmcode direkt auf dem Client-Rechner ausgeführt wird.
2. Ein Interpreter ist ein Programm, das einen als Quelltext vorliegenden Programmcode einliest und schrittweise in Maschinencode umwandelt und ausführen lässt. Interpreter sind immer plattformspezifisch, das heißt, es gibt jeweils eigene Interpreter für Microsoft/Intel, MacIntosh oder UNIX/Linux. Für JavaScript sind die Interpreter üblicherweise in die Browser integriert.
3. Der üblicher Weg zur Einbettung von Skriptcode in Webseite führt über das `<script>`-Tag, das je nach Aufgabe des Skriptcodes in den Header- oder den Body-Abschnitt der Webseite eingefügt wird.

```
<script type="text/javascript">
 . . . hier steht der Skript-Code . . .
</script>
```

- Um eine Skriptdatei zu laden, richtet man ein leeres `<script>`-Tag ein und gibt den URL der Skriptdatei als Wert des `src`-Attributs an.

```
<script type="text/javascript" src="farben.js"></script>
```

- JavaScript-Dateien werden mit der Extension `.js` abgespeichert.
- Das JavaScript-Objekt `document` repräsentiert die Webseite des aktuellen Browserfensters.
- Das JavaScript-Objekt `window` repräsentiert das aktuelle Browserfensters?

## Antworten zu den Übungen

- Siehe Kapitel 8.4.

## D.9 Tag 9: JavaScript-Grundkurs

### Antworten zum Quiz

- Folgende Variablennamen waren nicht korrekt:

```
var drei!; // Sonderzeichen ! in Variablenname
var 3fach; // Zahl am Anfang
var guter-Wert; // Sonderzeichen - in Variablenname
var Gärung; // Umlaute werden meist akzeptiert, man sollte
 // sie aber vermeiden
var new; // Schlüsselwort
```

- Für `wert1 = 13`, `wert2 = 4`, `wert3 = 1.5`, `wert4 = »23«` ergeben sich folgende Werte:

```
wert1 / wert2; // 3.25
wert1 % wert2; // 1
wert1 / wert3; // 8.6666
wert1 + wert4 * wert3; // 47.5
(wert1 + wert4) * wert3; // 1984.5
```

- Die Schleife wird endlos ausgeführt. Dies liegt daran, dass die `continue`-Anweisung vor dem Erhöhen der Schleifenvariable `loop` erfolgt. Wenn die `continue`-Anweisung das erste Mal ausgeführt wird (`loop` hat den Wert 9 erreicht), wird die aktuelle Schleifeniteration abgebrochen. Es beginnt eine neue Iteration, die fatalerweise mit dem gleichen `loop`-Wert arbeitet. Die Schleife führt erneut die `continue`-Anweisung aus, die Schleifenvariable wird wieder nicht verändert, es beginnt wieder eine neue Iteration, die wie die vorangehenden Iterationen aussieht.

```
var loop = 1;
var ergebnis = 0;
while(loop <= 10)
{
 ergebnis = loop*loop*loop;
 if (ergebnis > 700)
 continue;
 document.writeln(ergebnis);
 ++loop;
}
```

4. Zum Abspeichern und Bearbeiten einer Sammlung gleicher Daten verwendet man am besten ein Array-Objekt.
5. Zum Abspeichern und Bearbeiten zusammengehörender, aber unterschiedlicher Daten definiert man am besten eine Klasse.

## Antworten zu den Übungen

1. Wir implementieren die Passwortabfrage als Skriptcode im Header-Bereich und verzweigen je nachdem, ob das eingegebene Passwort korrekt ist oder nicht zur geheimen Webseite oder zu einer Seite mit einer Fehlermeldung. Auf diese Weise verhindern wir, dass der Besucher der Webseite sich in seinem Browser den Quelltext der Webseite mit der Passwortabfrage anzeigen lassen kann. Denn da wir das richtige Passwort nicht codiert haben und auch nicht aus einer serverseitigen Datenbank einlesen, sondern es uncodiert im Quelltext des Skriptcodes steht, könnte er die Passwortabfrage sonst leicht knacken.



*Aber auch die hier vorgestellte Methode ist kein Schutz gegen echte Hacker. Mit einem Seitenladeprogramm (wie wget unter Linux), können diese den Quellcode und damit das Passwort leicht einsehen.*

### Listing 4.12: password.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Passwort</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">

var eingabe;
eingabe = prompt("Geben Sie das Passwort ein: ", "");
if (eingabe == "geheim")
 {
 location.href = "meineSeite.html";
 }
else
 {
 location.href = "sorry.html";
 }
</script>
</head>
<body>
<h1>Passwortabfrage</h1>
</body>
</html>
```

2. Die korrespondierende while-Schleife sieht wie folgt aus:

```
var fahrenheit = -30;
var celsius = 0;
while (fahrenheit <= 90)
 {
```

```

celsius = (fahrenheit - 32) * 5 / 9;
document.write("<p>" + fahrenheit + " Fahrenheit entsprechen "
 + celsius + " Grad Celsius</p>");
fahrenheit += 10;
}

```

- Wie Sie bereits aus Kapitel 2.8 wissen, werden Farben im HTML-Code als RGB- Farben codiert, das heißt als eine Folge von 6 Hexadezimalwerten im Bereich zwischen 1 und F, die jeweils paarweise die Farbintensität für die drei Grundfarben Rot, Grün und Blau festlegen. Um zufällige Farben aus dem gesamten möglichen Wertebereich zu erhalten, kann man so vorgehen, dass man 6 zufällige Hexadezimalwerte zieht und zu einem Farbwert zusammensetzt.

Ein kleines Problem dabei ist, dass wir mit Hilfe der Funktion `Math.random()` und einer kleinen Umrechnung zwar Werte zwischen 1 und 16 erzeugen können, für die Farbangabe aber Hexadezimalwerte zwischen 1 und F benötigen. Wir lösen dieses Problem, indem wir ein Array mit Strings von »1« bis »F« anlegen und mit Hilfe von `Math.random()` Werte zwischen 0 und 15 erzeugen, die wir als Indizes in das Array verwenden.

#### Listing 4.13: farben.html

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>

 <title>Funktionen</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
 <script type="text/javascript">
function farbe_setzen()
 {
 var farbwerte = new Array("0","1","2","3","4","5","6","7","8","9",
 "A","B","C","D","E","F");

 var loop, index;
 var farbe = "#";
 for (loop = 0; loop < 6; ++loop)
 {
 index = parseInt(Math.random() * 16);
 farbe += farbwerte[index];
 }
 document.bgColor = farbe;
 }
 </script>
</head>
<body>
<p onmouseover="farbe_setzen()">Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text ... </p>
<p onmouseover="farbe_setzen()">Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text ... </p>
<p onmouseover="farbe_setzen()">Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text ... </p>
</body>
</html>

```

## D.10 Tag 10: Mit JavaScript auf HTML-Elemente zugreifen

### Antworten zum Quiz

1. Häufig verwendete JavaScript-Ereignisse sind: onblur, onload, onchange, onclick, onmousemove, onmouseover, onmouseout, onselect, onsubmit, onreset.
2. Im Internet Explorer kann man auf das globale event-Objekt über window.event zugreifen. Das auslösende Objekt ist in der Eigenschaft srcElement gespeichert, die Koordinaten des Mausklicks in den Eigenschaften offsetX und offsetY. Im Netscape Navigator übergibt man das Objekt als Argument an die Ereignisbehandlungsfunktionen. Das auslösende Objekt ist in der Eigenschaft target gespeichert, die Koordinaten des Mausklicks in den Eigenschaften pageX und pageY. (Achtung: offsetX, offsetY und pageX, pageY beziehen sich nicht auf das gleiche Koordinatensystem. offsetX, offsetY beziehen sich auf das auslösende Objekt, pageX, pageY beziehen sich auf die Webseite.)
3. Mit document.images[2] greifen Sie auf das dritte <img>-Element einer Webseite zu?
4. Mit der DOM-Methode getElementById() kann man direkt auf HTML-Elemente mit einer id-Kennung zugreifen. Die DOM-Methode getElementById() wird vom Internet Explorer 5 und dem Netscape 6-Browser unterstützt.
5. Im Internet Explorer und dem Netscape 6-Browser kann man über die Eigenschaft style auf die Stileigenschaften eines Objekts zugreifen.
6. Wenn Sie im JavaScript-Code bestimmen wollen, in welchem Browser die Webseite gerade angezeigt wird, testen Sie mit Hilfe von if-Anweisungen:
  - welche spezifischen Eigenschaften und Methoden der Browser unterstützt,
  - wie sein Name lautet (navigator.appName oder navigator.appCodeName)
  - welche Version er hat (navigator.appVersion)
  - welche Browsermaschine er verwendet (navigator.userAgent)

## Antworten zu den Übungen

1. Das folgende Skript zeigt eine Möglichkeit zur Realisierung grafischer Rollover- Schaltflächen.

Die verschiedenen Bilder für den Rollover-Effekt werden vorab beim Laden der Webseite geladen. Dadurch läuft der Rollover-Effekt später, wenn der Webbesucher seine Maus über die Schaltflächen bewegt, schneller ab. Gespeichert werden die Bilder in zwei Array-Objekten: einem Array-Objekt für die Schaltflächen ohne Effekt (schalterAus) und ein Array-Objekt für die Schaltflächen mit Effekt (schalterAn).

Wenn der Webbesucher die Maus über eine der Schaltflächen bewegt (onmouseover-Ereignis), wird der Index der Schaltfläche an die Funktion rollover\_an() übergeben und die Funktion weist dem img-Objekt das zugehörige Bild mit Effekt zu. Beachten Sie, dass wir den Index sowohl zum Adressieren des <img>-HTML-Elements als auch zum Auswählen des neu anzuzeigenden Bildes verwenden. Dies geht, weil es auf der Webseite außer den Schaltflächen keine weiteren Bilder gibt, so dass die Indizes für die Schaltflächen (document.images[]) und die Bilder (schalterAn, schalterAus) übereinstimmen. Ansonsten müsste man den Index für die Adressierung der Schaltflächen berechnen (beispielsweise document.images[3 + index], wenn die Webseite vor den Schaltflächen noch drei andere Bilder enthält) oder eine switch- Anweisung implementieren.

Wenn der Webbesucher die Maus von einer der Schaltflächen herunterbewegt (onmouseout-Ereignis), wird der Index der Schaltfläche an die Funktion rollover\_aus() übergeben und die Funktion weist dem img-Objekt das zugehörige Bild ohne Effekt zu.

### Listing 4.14: webseite.html - Webseite mit Rollover-Schaltflächen

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
 <title>Rollover-Effekt</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />
<script type="text/javascript">
```

```

var schalterAus = new Array();
 schalterAus[0] = new Image();
 schalterAus[0].src = "button1a.gif";
 schalterAus[1] = new Image();
 schalterAus[1].src = "button2a.gif";
 schalterAus[2] = new Image();
 schalterAus[2].src = "button3a.gif";

var schalterAn = new Array();
 schalterAn[0] = new Image();
 schalterAn[0].src = "button1b.gif";
 schalterAn[1] = new Image();
 schalterAn[1].src = "button2b.gif";
 schalterAn[2] = new Image();
 schalterAn[2].src = "button3b.gif";

function rollover_an(index)
 {
 document.images[index].src = schalterAn[index].src;
 }
function rollover_aus(index)
 {
 document.images[index].src = schalterAus[index].src;
 }
</script>
</head>
<body style="background-image: url('blaustreifen.jpg');
 background-repeat: repeat-y;
 margin-top: 30px">
<table border="0" cellpadding="0" cellspacing="20px">
 <tr>
 <td width="60"> </td>
 <td><a href="#" onmouseover="rollover_an(0)"
onmouseout="rollover_aus(0)"></td>
 </tr>
 <tr>
 <td width="60"> </td>
 <td><a href="#" onmouseover="rollover_an(1)"
onmouseout="rollover_aus(1)"></td>
 </tr>
 <tr>
 <td width="60"> </td>
 <td><a href="#" onmouseover="rollover_an(2)"
onmouseout="rollover_aus(2)"></td>
 </tr>
</table>
</body>
</html>

```

## D.11 Tag 11: JavaScript und Formulare

### Antworten zum Quiz

1. Das zweite Formular einer Webseite wird durch `document.forms[1]` repräsentiert. Denken Sie immer daran,

- dass die Indizierung von JavaScript-Arrays mit 0 beginnt.
- Die Eigenschaft value hat für unterschiedliche Steuerelemente unterschiedliche Bedeutung
    - für Textfelder enthält die Eigenschaft value den Inhalt (Text) des Textfeldes
    - für Schalter enthält die Eigenschaft value den Titel des Schalters
    - Optionsfelder nutzen die Eigenschaft value als eindeutige Kennung. So kann man die Optionsfelder einer Gruppe (die alle den gleichen name-Wert haben) unterscheiden.
  - Sollen die Daten, die der Webbesucher in ein Formular eingegeben hat, zur serverseitigen Bearbeitung an einen Webserver geschickt werden, muss man das Formular mit einem Submit-Schalter bestücken und im action-Attribut des <form>- Tags den Ziel-URL für die Daten angeben (optional kann man über method noch die Art der Datenübertragung festlegen (Standardeinstellung ist GET)).
  - Will man Formulareingaben vor der Übersendung an den Webserver noch clientseitig bearbeiten (beispielsweise um die Vollständigkeit der Daten zu überprüfen), muss man eine passende JavaScript-Funktion aufsetzen, die - je nachdem, ob die Daten tatsächlich verschickt oder doch verworfen werden sollen - true oder false zurückliefert und den Rückgabewert dieser Funktion dem onsubmit-Ereignis des Formulars zuweisen.

## Antworten zu den Übungen

- Eine mögliche Lösung könnte wie folgt aussehen.

### Listing 4.15: optionen.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Optionsfelder</title>
 <meta http-equiv="Content-Script-Type" content="text/javascript" />

<script type="text/javascript">
function farbe(f)
 {
 document.bgColor = f;
 }
</script>
</head>
<body>
<h1>Wählen Sie die Hintergrundfarbe</h1>
<form>
 <table border="0" cellspacing="5">
 <tr>
 <td><input type="radio" name="hintergrund" value="weiss"
 onclick="farbe('white')" />Weiss</td>
 <td><input type="radio" name="hintergrund" value="rot"
 onclick="farbe('red')" />Rot</td>
 <td><input type="radio" name="hintergrund" value="blau"
 onclick="farbe('blue')" />Blau</td>
 </tr>
 </table>
</form>
</body>
</html>
```

- Eine mögliche Lösung für die Webseite zur Umrechnung von DM in Euro könnte wie folgt aussehen.



A mit Hilfe der Anweisung

```
parent.frames[3]. location.href = "neueSeite.html";
```

eine neue Seite in Frame D laden.

## Antworten zu den Übungen

1. Wenn Sie passende Bilder von Comic-Figuren gefunden haben (einscannen oder Bild/ClipArt-Sammlungen entnehmen), müssen Sie die Bilder für die Verwendung in der Webseite präparieren.

Erzeugen Sie zuerst aus dem Bild der Comic-Figur jeweils eigene Bilder für Kopf und Rumpf.

Speichern Sie die Bilder als GIF-Dateien und geben Sie den Index der Hintergrundfarbe als Index der transparenten Farbe an (wie dies geht, hängt von dem verwendeten Grafikprogramm ab).

Vergrößern Sie die Bilder. Die Kopfbilder sollten 158 mal 148 Pixel groß sein, die Rumpfbilder 200 mal 235 Pixel. Um das eigentliche Bild sollte dabei noch genügend freier weißer Hintergrund sein, denn wir müssen das eigentliche Bild noch innerhalb des Bildrechtecks positionieren. (Sind Ihre Bilder zu groß, müssen Sie sie zuerst herabskalieren.)

Schneiden Sie aus dem Kopfbild den Kopf heraus und verschieben Sie den Kopf so innerhalb des Bildes, dass der Halsansatz an der Position 85, 125 zu liegen kommt. Schneiden Sie aus dem Rumpfbild den Rumpf heraus und verschieben Sie den Rumpf so innerhalb des Bildes, dass der Halsansatz an der Position 90, 100 zu liegen kommt. So wird sichergestellt, dass die Köpfe stets ordentlich am Rumpf ansetzen. (Die oben genannten Werte gelten allerdings nur, wenn Sie die Angaben für die absolute Positionierung im HTML-Code der Webseite *hauptseite.html* nicht verändern.)

## D.13 Tag 13: Cookies

### Antworten zum Quiz

1. Das JavaScript-Objekt `document.cookies` repräsentiert alle Cookies, die für eine Webseite gesetzt wurden.
2. Um einen Cookie für eine Webseite zu setzen, erzeugt man zuerst einen String für den Cookie. Ein solcher Cookie-String muss zumindest aus einem `name=WERT`-Paar (und gegebenenfalls einem `expires=DATUM`-Paar) bestehen. Diesen String weist man dann dem `document.cookie`-Objekt zu.

```
var str = "meinCookie=" + escape("Inhalt des Cookies")
 + "; expires=Fri, 12 Jul 2004 12:34:55 GMT";
document.cookie = str;
```

3. Um den Inhalt eines Cookies abzufragen, geht man den Inhalt des `document.cookie`-Objekts durch, bis man auf den Namen des Cookies trifft. Dann extrahiert man den Wert des Cookies, decodiert die Sonderzeichen und liefert das Ergebnis zurück.
4. Wenn man einen Cookie für mehrere Webseiten setzen möchte, muss man als `path`-Attribut des Cookies ein Verzeichnis angeben, das den betreffenden Webseiten übergeordnet ist.

### Antworten zu den Übungen

1. Zuerst erweitert man das Formular um eine Gruppe von zwei Optionsfelder (gleiche `name`-Werte).

**Listing 4.17: Auszug aus `formulare.html`**



```

while (beginn < naechster)
{
ende = document.cookie.indexOf("|", beginn);
if (ende == -1)
 ende = naechster;
if(i == 2 || i == 3)
{
var gesetzt = unescape(document.cookie.substring(beginn,
 ende));

switch(gesetzt)
{
case "true": document.forms[0].elements[i].checked = true;
 break;
case "false": document.forms[0].elements[i].checked = false;
 break;
}
}
else
 document.forms[0].elements[i].value =
 unescape(document.cookie.substring(beginn, ende));
beginn = ende + 1;
i += 1;
}
return 1;
}

```

Neu hinzugekommen ist der mittlere Teil. Hier wird anhand des Index *i* überprüft, ob gerade der Wert für eines der Optionsfelder bearbeitet wird. Wenn ja, müssen wir statt der Steuerelement-Eigenschaft *value* die Eigenschaft *checked* setzen. Dabei ist zu beachten, dass uns die Methoden *substring()* und *unescape()* einen String zurückliefern. In einer *switch*-Verzweigung prüfen wir, ob der String "true" oder "false" lautet, und setzen entsprechend das Steuerelement auf *true* oder *false*.

## D.14 Tag 14: Erste serverseitige Techniken

### Antworten zum Quiz

1. Perl, C++, CGI, ASP, PHP.
2. Um das Datum der letzten Änderung einer bestimmten Datei in eine Webseite einzufügen, verwendet man den SSI-Befehl `#flastmod`.

```
<!--#flastmod file="datei.txt" -->
```

3. Aus zwei Gründen sollte man einen (öffentlichen) Webserver nicht so konfigurieren, dass er alle Webseiten mit der Extension `.html` nach Server Side Includes durchsucht:

Das Durchsuchen von HTML-Dokumente nach Server Side Includes setzt die Performance des Webserver herab. Je mehr Dateien der Webserver dabei untersuchen muss, um so stärker wird seine Leistung herabgesetzt. Nun brauchen moderne Webserver zum Durchsuchen der Webseiten und zur Verarbeitung der Server Side Includes nicht allzu lange, doch sollte man den Webserver nicht unnötig Webseiten analysieren lassen (wie es meist der Fall ist, wenn man als Extension für die zu analysierenden Seiten `.html` wählt).

Zudem entsteht ein nicht zu unterschätzendes Sicherheitsloch, wenn der Webserver HTML-Seiten analysiert, die Formulareingaben von Websurfern enthalten (siehe Frage/Antwort-Teil)

## Antworten zu den Übungen

1. Eine mögliche Lösung für die Webseite mit der geforderten Datumsausgabe könnte wie folgt aussehen.

### Listing 4.20: lastmodified.shtml mit formatierter Datumsausgabe

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Server Side Includes</title>
</head>
<body>
<h1>Das Wetter</h1>
<p>Morgens diesig, in den Niederungen dichter Nebel. Tagsüber Temperaturen
zwischen 4 und 6 Grad Celsius. Gegen Abend Gewitter. Fazit: Wer heute Sonne
haben will, muss sie im Herzen tragen.
<!--#config timefmt="%A, der %d.%m.%y (%H:%M:%S)" -->
<p>Letzte Änderung: <!--#flastmod file="lastmodified.shtml" --></p>
</body>
</html>
```

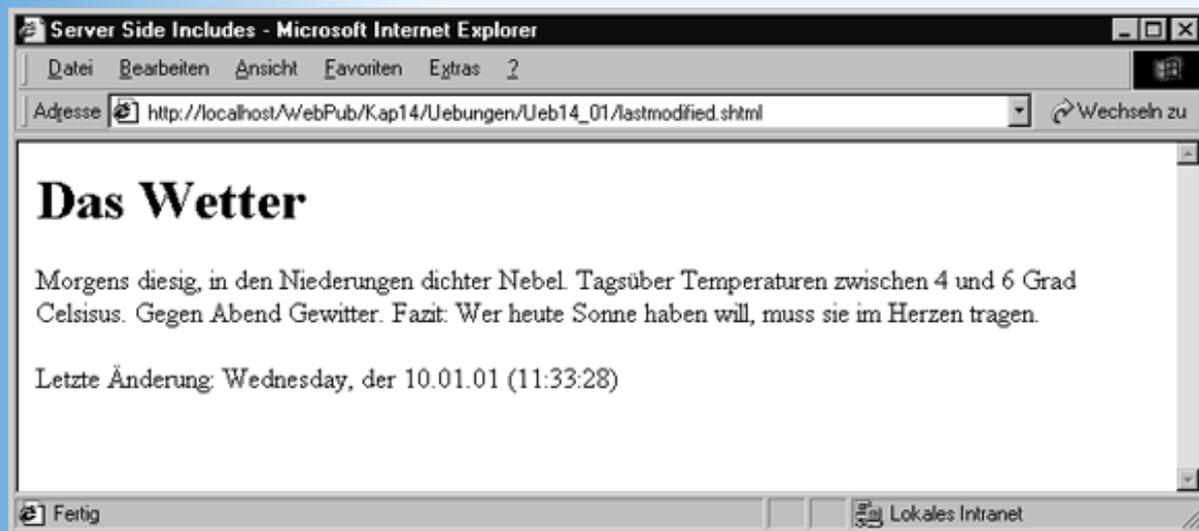


Abbildung D.7: Formatierte Datumsausgabe

## D.15 Tag 15: Java-Applets

### Antworten zum Quiz

1. Mit Java kann man nicht nur Applets für Webseiten erstellen. Java ist eine Sprache für professionelle Programmierer, mit der man jede Art von Programm erstellen kann.
2. Applets werden in Form von Bytecode über das Netz übertragen. Dieser Bytecode wird beim Laden in einen Java-fähigen Browser von diesem interpretiert. So können Applets auf beliebigen Plattformen ausgeführt werden - einzige Voraussetzung ist, dass auf dem jeweiligen Rechner ein Java-fähiger Browser installiert ist.
3. Wenn Sie ein Applet mit Hilfe des <applet>-Tags in eine Webseite einbetten, müssen Sie den Namen des Applets (code-Attribut) sowie die Breite und Höhe des Anzeigebereichs für das Applet (width- und height-Attribut) angeben. Optional können Sie das Verzeichnis des Applets angeben (codebase-Attribut) und/oder Parameter an das Applet übergeben.
4. Die Hauptklasse eines Applets muss von der Basisklasse Applet abgeleitet werden.
5. In den Anzeigebereich eines Applets kann man nicht schreiben, sondern nur zeichnen. Textausgaben werden daher unter Verwendung der Graphics-Methode drawString(text, x, y) in das Applet »gezeichnet«.

6. Falls Sie den Aufruf in *Mausklick.java* übersehen haben: die Hintergrundfarbe eines Applets wird durch Aufruf der Methode `setBackground()` gesetzt.

## Antworten zu den Übungen

1. Um das Fleckenapplet so zu überarbeiten, dass beim Klicken in das Applet die Farbe der Flecken neu festgelegt wird, sind drei Änderungen nötig:

Sie müssen die Adapter-Klassen für die Ereignisbehandlung importieren:

```
import java.awt.event.*;
```

Sie müssen eine eigene Lauscherklasse von `MouseListener` ableiten und in dieser die Methode `mouseClicked()` überschreiben:

```
class MeinMausLauscher extends MouseAdapter
{
 public void mouseClicked(MouseEvent e)
 {
 m_farbe = new Color((int) (255*Math.random()),
 (int) (255*Math.random()),
 (int) (255*Math.random()));

 repaint();
 }
}
```

Sie müssen Ihre Lauscherklasse als Ereignisbehandlungsmethode für das Mausclick- Ereignis des Applets registrieren:

```
public void init()
{
 ...
 addMouseListener(new MeinMausLauscher());
}
```

## D.16 Tag 16: Animationen

### Antworten zum Quiz

1. Eine für alle Plattformen verfügbare Grafik-Software, mit der man auch GIF- Animationen erstellen kann, ist die ImageMagick-Programmsammlung, die man von [www.simplesystems.org/ImageMagick](http://www.simplesystems.org/ImageMagick) herunterladen kann!
2. Um eine JavaScript-Funktion in regelmäßigen Zeitabständen immer wieder ausführen zu lassen, braucht man nur am Ende der Funktion die JavaScript-Methode `setTimeout()`, mit dem Funktionsnamen als erstem und der Zeitdauer als zweitem Argument aufzurufen.
3. Threads werden in Java mit Hilfe der Klasse `Thread` erzeugt - entweder durch Erzeugung eines `Thread`-Unterobjekts oder durch Ableitung der Hauptklasse von `Thread`. Java-Applets kann man auch dadurch threadfähig machen, dass man die Applet-Klasse die Schnittstelle `Runnable` implementieren lässt, die Methode `run()` definiert, und dann ein `Thread`-Objekt erzeugt und startet.
4. Bild-für-Bild-Animationen, Motion Tweening, Shape Tweening, Motion Tweening entlang eines Laufpfades.
5. Flash-Animationen werden durch eine Verschachtelung eines `<object>`- und eines `<embed>`-Tags in Webseiten eingebettet. Das `<object>`-Tag erzeugt die Animation im Internet Explorer, das `<embed>`-Tags in den Netscape-Browsern.

## D.17 Tag 17: CGI und Perl

## Antworten zum Quiz

1. Skalare Variable haben vor dem eigentlichen Namen immer ein \$, Array-Variablen ein @.
2. In Perl-Strings in doppelten Anführungszeichen werden Variablennamen durch die Werte der Variablen ersetzt, in Strings in einfachen Anführungszeichen nicht.
3. Zahlenwerte vergleicht man mit ==, !=, < etc. - die Antwort lautet also \$var1 < \$var2
4. Strings vergleicht man mit eq, ne, lt etc. - die Antwort lautet also \$str1 lt \$str2
5. Lösungen zu allgemeinen Programmieraufgaben sucht man am Besten im CPAN.
6. Die Perl-CGI-Methode header gibt den HTTP-Header zum nachfolgenden HTML- Code einer dynamisch erzeugten Webseite aus.
7. Will man zusammen mit den Formulareingaben weitere Daten an ein CGI-Programm schicken, richtet man im Formular verborgene Felder (<input type="hidden" />) ein.
8. Will ein CGI-Programm statt einer dynamisch generierten Webseite den URL einer bestehenden Webseite als Antwort an den Browser zurückliefern, schickt er einen Location-Header.

## Antworten zu den Übungen

1. Eine mögliche Lösung könnte wie folgt aussehen. Die Anzahl der Pro-Meldungen, der Kontra-Meldungen und die Gesamtzahl der Meldungen werden jede für sich in einer Zeile einer Datei namens *umfrage.dat* im CGI-Verzeichnis des Servers abgespeichert.

### Listing 4.21: umfrage.dat

```
200
100
300
```

Dann könnte das CGI-Programm wie folgt implementiert werden:

### Listing 4.22: umfrage.pl

```
#!/usr/bin/perl -w
use CGI qw(:standard);
use strict;
my $pro;
my $kontra;
my $gesamt;
my $proProzent;
my $kontraProzent;
***** Umfragestatistik einlesen *****
my $statistik = "umfrage.dat"; # Datei muss in gleichem Verzeichnis
 # stehen wie das CGI-Programm
open(STAT, "$statistik")
or
die "\nDatei $statistik konnte nicht zum Lesen geoeffnet werden\n";
$pro = <STAT>; # erste Zeile lesen
chomp($pro); # Zeilenumbruch entfernen
$kontra = <STAT>; # zweite Zeile lesen
chomp($kontra); # Zeilenumbruch entfernen
$gesamt = <STAT>; # dritte Zeile lesen
chomp($gesamt); # Zeilenumbruch entfernen
close(STAT);

***** Eingaben aus Formular verarbeiten *****
my $option = param('R1');
if ($option eq "pro") # pro-Option gesetzt
```

```

 {
 ++$pro;
 }
 else # kontra-Option gesetzt
 {
 ++$kontra;
 }
++$gesamt;
Prozentwerte berechnen
$proProzent = $pro * 100 / $gesamt;
$kontraProzent = $kontra * 100 / $gesamt;
***** Neuen Stand speichern *****
open(STAT, "> $statistik")
 or
 die "\nDatei $statistik konnte nicht zum Schreiben geoeffnet werden\n";
print STAT $pro, "\n";
print STAT $kontra, "\n";
print STAT $gesamt, "\n";
close(STAT);
***** Umfragestatistik zurücksenden *****
my $antwort = <<HERE_UMFRAGE;
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
 <title>Umfrage zu internetfreiem Sonntag</title>
</head>
<body>
<h1>Aktueller Stand der Umfrage</h1>
<p>Sind Sie für den internetfreien Sonntag?</p>
<table border="0" width="250" cellspacing="5" cellpadding="5">
 <tr>
 <td width="50" align="right">Dafür :</td>
 <td width="200">$proProzent %</td>
 </tr>
 <tr>
 <td width="50" align="right">Dagegen :</td>
 <td width="200" align="left">$kontraProzent %</td>
 </tr>
</table>
</body>
</html>
HERE_UMFRAGE
print header, $antwort;

```



*Wenn Sie unter UNIX/Linux arbeiten, müssen Sie die Zugriffsrechte der Dateien korrekt setzen. Das Perl-Skript muss vom Webserver ausführbar sein (beispielsweise `chmod 755 umfrage.pl`), die Datei `umfrage.dat` muss von dem Perl-Skript gelesen und geschrieben werden können (beispielsweise `chmod 666 umfrage.dat`).*

## D.18 Tag 18: Microsoft präsentiert... ASP

## Antworten zum Quiz

1. Nein, ASP ist keine Programmiersprache, sondern eher eine Technologie. Auf ASP- Seiten können Sie mehrere Programmiersprachen verwenden, am weitesten verbreitet sind VBScript und JScript.
2. Mit Option Explicit erzwingen Sie, dass alle verwendeten Variablen zuvor mit Dim geklariert werden müssen. Zum einen müssen Sie dann sauberer und konzentrierter programmieren, aber Tippfehler bei Variablennamen werden Ihnen in der Regel direkt vom ASP-Interpreter gemeldet.
3. Im Gegensatz zu JavaScript - beides. Während bei JavaScript Vergleiche noch mit == angestellt werden mussten, erfüllt das Gleichheitszeichen bei VBScript eine Doppelfunktion.
4. Bei Select Case können Sie einen Ausdruck (z.B. eine Variablen) auf mehrere verschiedene Werte überprüfen. Bei If-Then können Sie einen Ausdruck auf einen Wert überprüfen; bei der Verwendung von Else/Elseif können Sie jedoch andere Ausdrücke überprüfen, sind also nicht auf einen Ausdruck festgelegt.
5. Entweder mit For (die Zählvariable läuft von 0 bis UBound(Arrayname)), oder - noch bequemer - mit For-Each.
6. Dann wird das Formular auf dieselbe Seite verschickt. Ist das Formular selbst eine ASP-Seite, oder wollen Sie das Formular ggf. erneut anzeigen (etwa wenn es nicht vollständig ausgefüllt worden ist), ist das sehr praktisch.
7. Greifen Sie mit Request("name-Attribut") die Werte als Kollektion ab, und durchschreiten Sie diese Kollektion mit For-Each.
8. Der zweite Parameter gibt an, in welchem Modus die Datei geöffnet werden soll. Im Modus 8 wird die Datei im Anhängen-Modus geöffnet, das heißt neue Daten werden ans Ende der bestehenden Datei geschrieben, der bisherige Dateiinhalt geht also nicht verloren. Im Modus 2 wird der bestehende Dateiinhalt gelöscht bzw. durch die neuen Inhalte überschrieben. Um genau zu sein, gibt es sogar noch einen dritten Modus, 1, der zum Lesen aus einer Datei dient.
9. Beim Lesen von Cookies müssen Sie auf Request.Cookies zugreifen, und können das überall auf der ASP-Seite machen. Zum Schreiben von Cookies müssen Sie Response.Cookies verwenden, und müssen dies darüber hinaus auch noch vor dem ersten HTML-Code machen.

## Antworten zu den Übungen

1. Die aktuelle Stunde wird mit Hour(Now) ausgelesen, der Rest sind einfache Fallunterscheidungen:

### Listing 4.23: uebung1.asp

```
<%
Option Explicit
Dim intStunde, strHallo
intStunde = Hour(Now)
If intStunde >= 6 And intStunde <=10 Then
 strHallo = "Guten Morgen!"
ElseIf intStunde >=11 And intStunde <=13 Then
 strHallo = "Mahlzeit!"
ElseIf intStunde >=14 And intStunde <=17 Then
 strHallo = "Guten Tag!"
ElseIf intStunde >=18 And intStunde <=21 Then
 strHallo = "Guten Abend!"
Else
 strHallo = "Gute Nacht!"
End If
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Begrüßung</title>
</head>
```

```
<body bgcolor="white">
<h2><% =strHallo %></h2>
</body>
</html>
```

- Der Trick besteht in der Regel darin, das value-Attribut der entsprechenden Formularelemente mit den Daten aus Request.Form zu füllen. Ausnahmen sind Checkboxes, Radiobuttons und Auswahllisten, bei denen ein selected oder checked eingefügt werden muss. Im folgenden Listing werden die Formulardaten auf Vollständigkeit überprüft, und im Ernstfall eine Warnmeldung ausgegeben sowie das Formular wieder angezeigt, mit den bereits eingegebenen Werten vorausgefüllt:

#### Listing 4.24: uebung2.asp

```
<%
Option Explicit
Dim bVollstaendig, i
bVollstaendig = False
If Request.Form("submit")="Abschicken" Then
 bVollstaendig = True
 If Request.Form("Geschlecht") = "" Then
 bVollstaendig = False
 End If
 If Request.Form("Vorname") = "" Then
 bVollstaendig = False
 End If
 If Request.Form("Nachname") = "" Then
 bVollstaendig = False
 End If
 If Request.Form("Windows")="" And Request.Form("Linux")="" Then
 bVollstaendig = False
 End If
End If
If bVollstaendig = True Then
 Response.Redirect "danke.asp"
End If
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Formulare</title>
</head>
<body bgcolor="white">
<h3>Formularversand</h3>
<p>
<%
 If Not bVollstaendig Then
 Response.Write "Bitte füllen Sie alle Felder aus!"
 Response.Write "</p><p>"
 End If
%>
<form method="post">
<input type="text" name="Vorname" size="20"
value="<% =Request.Form("Vorname") %>" />Vorname

<input type="text" name="Nachname" size="20"
value="<% =Request.Form("Nachname") %>" />Nachname

<input type="radio" name="Geschlecht" value="m"
```

```

<% If Request.Form("Geschlecht") = "m" Then %> checked<% End If %>
>mänlich
<input type="radio" name="Geschlecht" value="f"
<% If Request.Form("Geschlecht") = "f" Then %> checked<% End If %>
>weiblich

Ich nutze
<input type="checkbox" name="Windows" value="ja"
<% If Request.Form("Windows") = "ja" Then %> checked<% End If %>
>Windows
<input type="checkbox" name="Linux" value="ja"
<% If Request.Form("Linux") = "ja" Then %> checked<% End If %>
>Linux

<select name="Tag" size="1">
<%
 For i=1 To 21
 %>
<option value="<% =i %>"
<% If Request.Form("Tag") = i&" Then %> selected<% End If %>
><% =i %></option>
<%
 Next
 %>
</select>Dieser Tag gefällt mir am Besten

<input type="submit" name="submit" value="Abschicken" />
</form>
</p>
</body>
</html>

```



*Beachten Sie, wie wir bei der Vorauffüllung des Pulldown-Menüs die Variable i in einen String umwandeln mussten!*

- Der Wert des Cookies wird überprüft. Sofern dieser leer ist (oder nur aus spitzen öffnenden und schließenden Klammern besteht), wird das Formular ausgegeben, ansonsten die Hintergrundfarbe. Am Anfang des Skripts muss noch der Fall abgefangen werden, dass das Formular gerade verschickt worden ist, und der Cookie entsprechend gesetzt werden. Da der Cookie erst mit der Seite an den Client geschickt wird, kann in diesem Fall noch nicht direkt darauf zugegriffen werden; die Lieblingsfarbe muss also in einer Variablen zwischengespeichert werden

#### **Listing 4.25: uebung3.asp**

```

<%
Option Explicit
Dim strFarbe
strFarbe = ""

If Request.Form("submit") = "Abschicken" Then
 Response.Cookies("Farbe") = Request.Form("Farbe")
 Response.Cookies("Farbe").Expires = Date + 60
 strFarbe = Request.Form("Farbe")
Else
 strFarbe = Request.Cookies("Farbe")

```

```

End If

strFarbe = Replace(strFarbe, "<", "")
strFarbe = Replace(strFarbe, ">", "")
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Cookies</title>
</head>
<%
 If strFarbe <> "" Then
%>
<body bgcolor="<% =strFarbe %>">
<%
 Else
%>
<body bgcolor="white">
<%
 End If
%>
<h2>Personalisierung</h2>
<p>
<%
 If strFarbe = "" Then
%>
<form method="post">
<input type="text" name="Farbe" size="20" /> Lieblingsfarbe

<input type="submit" name="submit" value="Abschicken">
</form>
<%
 Else
%>
<h2>Willkommen auf Ihrer personalisierten Seite!</h2>
<%
 End If
%>
</p>
</body>
</html>

```

## D.19 Tag 19: PHP, der Shooting-Star

### Antworten zum Quiz

1. Zend ist die Engine, die hinter PHP 4 steckt. Ihr Name bildet sich aus den Entwicklern der neuen PHP-Version. Rasmus ist der ursprüngliche Entwickler von PHP und hilft immer noch bei der Weiterentwicklung.
2. Das Gleichheitszeichen ist in PHP nur ein Zuweisungsoperator. Als Vergleichsoperator fungiert ==.
3. Bei switch können Sie einen Ausdruck (z.B. eine Variablen) auf mehrere verschiedene Werte überprüfen. Bei if können Sie einen Ausdruck auf einen Wert überprüfen; bei der Verwendung von else/elseif können Sie jedoch andere Ausdrücke überprüfen, und sind also nicht auf einen Ausdruck festgelegt.
4. Entweder mit for (die Zählvariable läuft von 0 bis count(\$Array)), oder - noch bequemer - mit foreach.
5. Die Spezialvariable \$PHP\_SELF enthält die URL des aktuellen PHP-Skripts. Viele Leute setzen das action-Attribut des Formulars auf diese Variable, was aber völlig unnötig ist.
6. Im HTML-Code muss das name-Attribut der Auswahlliste auf [] enden. Um auf alle Elemente zuzugreifen,

verwenden Sie eine for-Schleife über `HTTP_POST_VARS["name-Attribut"]`. Verwenden Sie dabei numerische Indizes, also etwa `HTTP_POST_VARS["name-Attribut"][0]`.

- Der zweite Parameter gibt an, in welchem Modus die Datei geöffnet werden soll. Im Modus "a" wird die Datei im Anhängen-Modus geöffnet, das heißt, neue Daten werden ans Ende der bestehenden Datei geschrieben, und der bisherige Dateiinhalt geht nicht verloren. Im Modus »w« wird der bestehende Dateiinhalt gelöscht bzw. durch die neuen Inhalte überschrieben. Um genau zu sein, gibt es sogar noch einen dritten Modus, "r", der zum Lesen aus einer Datei dient. Und für jeden dieser drei Modi gibt es noch eine Spezialvariante, das soll aber nicht mehr Thema des heutigen Tages sein.
- Nur der erste Parameter, der den Namen des Cookies angibt.

## Antworten zu den Übungen

- Die aktuelle Stunde wird mit im von `getdate` zurückgegebenen Array über den Schlüssel "hours" ausgelesen, der Rest sind einfache Fallunterscheidungen:

### Listing 4.26: uebung1.php

```
<?php
 $datum = getdate();
 $stunde = $datum["hours"];
 if ($stunde >= 6 && $stunde <=10) {
 $hallo = "Guten Morgen!";
 } elseif ($stunde >= 11 && $stunde <= 13) {
 $hallo = "Mahlzeit!";
 } elseif ($stunde >= 14 && $stunde <= 17) {
 $hallo = "Guten Tag!";
 } elseif ($stunde >= 18 && $stunde <= 21) {
 $hallo = "Guten Abend!";
 } else {
 $hallo = "Gute Nacht!";
 }
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Begrüßung</title>
</head>
<body bgcolor="white">
<h2><?=$hallo ?></h2>
</body>
</html>
```

- Der Trick besteht in der Regel darin, das `value`-Attribut der entsprechenden Formularelemente mit den Daten aus `$HTTP_POST_VARS` zu füllen. Ausnahmen sind Checkboxes, Radiobuttons und Auswahllisten, bei denen ein `selected` oder `checked` eingefügt werden muss. Im folgenden Listing werden die Formulardaten auf Vollständigkeit überprüft. . Im Ernstfall wird eine Warnmeldung ausgegeben und das mit den bereits eingegebenen Werten wieder angezeigt:

### Listing 4.27: uebung2.php

```
<?php
 $vollstaendig = false;
 if ($HTTP_POST_VARS["submit"] == "Abschicken") {
 $vollstaendig = true;
 if (!isset($HTTP_POST_VARS["Geschlecht"])) {
```

```

 $vollstaendig = false;
}
if (!isset($_HTTP_POST_VARS["Vorname"])) {
 $vollstaendig = false;
}
if (!isset($_HTTP_POST_VARS["Nachname"])) {
 $vollstaendig = false;
}
if (!isset($_HTTP_POST_VARS["Windows"]) &&
 !isset($_HTTP_POST_VARS["Linux"])) {
 $vollstaendig = false;
}
if ($vollstaendig) {
 header("Location: danke.php");
}
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Formulare</title>
</head>
<body bgcolor="white">
<h3>Formularversand</h3>
<p>
<?php
 if (!$vollstaendig) :
?>
Bitte füllen Sie alle Felder aus!</p>
<p>
<?php
 endif;
?>
<form method="post">
<input type="text" name="Vorname" size="20"
 value="<?php print $_HTTP_POST_VARS["Vorname"]; ?>" />Vorname

<input type="text" name="Nachname" size="20"
 value="<?php print $_HTTP_POST_VARS["Nachname"]; ?>" />Nachname

<input type="radio" name="Geschlecht" value="m"
?>
 if ($_HTTP_POST_VARS["Geschlecht"] == "m") {
 print " checked";
 }
?>
>männlich
<input type="radio" name="Geschlecht" value="f"
?>
 if ($_HTTP_POST_VARS["Geschlecht"] == "f") {
 print " checked";
 }
?>
>weiblich

Ich nutze
<input type="checkbox" name="Windows" value="ja"
?>
 if ($_HTTP_POST_VARS["Windows"] == "ja") {

```

```

 print " checked";
 }
?>
>Windows
<input type="checkbox" name="Linux" value="ja"
<?php
 if ($_HTTP_POST_VARS["Linux"] == "ja") {
 print " checked";
 }
?>
>Linux

<select name="Tag" size="1">
<?php
 for ($i=1; $i<=21; $i++) :
?>
<option value="<?=$i ?>"
<?php
 if ($_HTTP_POST_VARS["Tag"] == $i) {
 print " selected";
 }
?>
><?=$i ?></option>
<?php
 endfor;
?>
</select>Dieser Tag gefället mir am Besten

<input type="submit" name="submit" value="Abschicken" />
</form>
</p>
</body>
</html>

```

- Der Wert des Cookies wird überprüft. Sofern dieser leer ist (oder nur aus spitzen öffnenden und schließenden Klammern besteht), wird das Formular ausgegeben, ansonsten die Hintergrundfarbe. Am Anfang des Skripts muss noch der Fall abgefangen werden, dass das Formular gerade verschickt worden ist, und der Cookie entsprechend gesetzt werden. Da der Cookie erst mit der Seite an den Client geschickt wird, kann in diesem Fall noch nicht direkt darauf zugegriffen werden; die Lieblingsfarbe muss also in einer Variablen zwischengespeichert werden

#### Listing 4.28: uebung3.php

```

<?php
 $farbe = "";

 if (isset($_HTTP_POST_VARS["submit"])) {
 setcookie("Farbe", $_HTTP_POST_VARS["Farbe"], time()+60*60*24*60);
 $farbe = $_HTTP_POST_VARS["Farbe"];
 } else {
 $farbe = $_HTTP_COOKIE_VARS["Farbe"];
 }

 $farbe = str_replace("<", "", $farbe);
 $farbe = str_replace(">", "", $farbe);
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN
 http://www.w3.org/TR/html4/strict.dtd">
<html>

```

```

<head>
<title>Cookies</title>
</head>
<?php
 if ($farbe != "") :
?>
<body bgcolor="<?=$farbe ?>">
<?php
 else :
?>
<body bgcolor="white">
<?php
 endif;
?>
<h2>Personalisierung</h2>
<p>
<?php
 if ($farbe = "") :
?>
<form method="post">
<input type="text" name="Farbe" size="20" /> Lieblingsfarbe

<input type="submit" name="submit" value="Abschicken">
</form>
<?php
 else :
?>
<h2>Willkommen auf Ihrer personalisierten Seite!</h2>
<?php
 endif;
?>
</p>
</body>
</html>

```

## D.20 Tag 20: Datenbankanbindung

### Antworten zum Quiz

1. Beispielsweise CREATE TABLE, SELECT, INSERT, UPDATE, DELETE, DROP TABLE
2. Ein Autowert ist ein numerischer Datentyp, der sich bei jedem neuen Datenbankeintrag um 1 erhöht.
3. Die meisten Datenbanken bieten eine grafische Benutzeroberfläche (GUI) an. Alternativ können Sie auch CREATE TABLE und SQL verwenden.
4. Wenn Sie ODBC verwenden, können Sie über den DSN-Eintrag die Datenbank ansprechen. Unter ASP können Sie bei Verwendung von Access auch direkt den Namen der Access-Datei angeben.

### Antworten zu den Übungen

1. Die naheliegendste Möglichkeit besteht darin, die Tabelle um ein Datumsfeld zu erweitern. Dieses Datumsfeld wird beim Schreiben in die Datenbank mit dem aktuellen Tagesdatum gefüllt. Das SQL-Statement zum Lesen aus der Datenbank lautet dann SELECT \* FROM gaestebuch ORDER BY datum DESC. Alternativ dazu können Sie aber auch die Tabelle um einen Primärschlüssel erweitern, der gleichzeitig ein Autowert ist. Es ist nun klar, dass der zuletzt in die Datenbank geschriebene Wert die größte ID hat. Mit SELECT \* FROM gaestebuch ORDER BY id DESC erhalten Sie die Daten in der korrekten Reihenfolge.

## D.21 Tag 21: XML und XHTML

## Antworten zum Quiz

1. Ja.
2. XML-Dokumente beginnen mit
  - der XML-Deklaration, die das Dokument als XML-Dokument kennzeichnet
  - der optionalen Angabe einer Dokumenttypdeklaration (DTD)
  - dem Wurzelement des Dokuments (für XHTML-Dokumente lautet das Wurzelement <html>)
3. Man beendet das Tag mit der Zeichenfolge » />«.

## Antworten zu den Übungen

1. Die XHTML-Version der Webseite sollte ungefähr wie folgt aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "DTD/xhtml1-strict.dtd">
<html>
<head>
 <title>XHTML-Beispiel</title>
 <link rel="StyleSheet" type="text/css" href="sonne.css" />
</head>
<body>
<h1>Erde und Sonne</h1>
<div class="sonne"></div>
<div class="erde"></div>
</body>
</html>
```

Mit dem zugehörigen externen Stylesheet *sonne.css*:

```
body { background-color: black; color: white;
 margin-top: 10px}
div.sonne { position: absolute; left: 350px; top: 120px}
div.erde { position: absolute; left: 335px; top: 200px }
```

Damit Designs, wie das aus Übung 2.2, fehlerfrei betrachtet werden können, ist unbedingt auf pixelgenaues Arbeiten zu achten.

# Tag Anhang E

## Adressen

Hier einige hilfreiche Webadressen und Buchempfehlungen.

## E.1 HTML

- <http://www.w3.org/TR/1999/REC-html401-19991224> - die HTML 4-Spezifikation
- <http://www.w3.org/TR/1999/REC-CSS1-19990111> - die CSS1-Stylesheet-Spezifikation
- <http://www.w3.org/TR/1998/REC-CSS2-19980512> - die CSS2-Stylesheet-Spezifikation
- <http://www.w3.org/TR/2000/REC-xhtml1-20000126> - die XHTML 1-Spezifikation

## E.2 JavaScript

- <http://www.ecma.ch> - die ECMAScript-Spezifikation
- <http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001> - die DOM1-Spezifikation
- <http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113> - die DOM2-Spezifikation
- <http://www.teamone.de/selfhtml> - JavaScript-Tutorial
- <http://www.webreference.com> - Tipps und Tricks zu JavaScript und andere Technologien
- »JavaScript - Browserübergreifende Lösungen« von Christian Wenz, Galileo Press

## E.3 Java

- <http://java.sun.com> - die Java-Entwicklungsumgebung
- <http://www.borland.com> - die JBuilder-Entwicklungsumgebung zur Java-Programmierung
- »Jetzt lerne ich Java« von Dirk Louis, Peter Müller, Markt+Technik-Verlag
- »Jetzt lerne ich JBuilder« von Dirk Louis, Peter Müller, Markt+Technik-Verlag

- »Java 2 in 21 Tagen« von Laura Lemay, Markt+Technik-Verlag
- »Just Java and Beyond 1.1« von Peter van der Linden, Prentice Hall-Verlag.

## E.4 Perl

- <http://www.activestate.com/ActivePerl> - Perl-Interpreter (auch für Windows)
- <http://www.perl.com> - Perl-Interpreter
- »Jetzt lerne ich Perl« von Dirk Louis, Markt+Technik-Verlag
- »Perl« von Farid Hajji, Addison-Wesley-Verlag

## E.5 CGI

- »Jetzt lerne ich CGI-Programmierung« von Rafe Colburn, Markt+Technik-Verlag

## E.6 ASP

- »Jetzt lerne ich ASP« von Christian Wenz, Christian Trennhaus und Andreas Kordwig, Markt+Technik-Verlag

## E.7 PHP

- <http://www.php.net> - PHP4 Win32-Binär und Quellcodedateien
- »PHP 4« von Egon Schmid, Christian Cartus, Wolfgang Drews, Hartmut Holzgraefe, Uwe Steinmann und Christian Wenz, Markt+Technik-Verlag

## E.8 XML

- <http://www.w3.org/TR/2000/REC-xml-20001006> - die XML 1-Spezifikation
- »XHTML by example« von Ann Navarro, Que-Verlag
- »XML in 21 Tagen« von Simon North und Paul Hermans, Markt+Technik-Verlag

## E.9 Software

- <http://httpd.apache.org> - der Apache-Server
- <http://www.redhat.com> - RedHat-Website, von der man den Apache-Server als RPM- Paket herunterladen kann
- <http://www.omnicron.ab.ca> - der OmniHTTPd-Server für Windows
- <http://www.xitami.com> - der Xitami-Server für Windows und Linux
- <http://www.ipswitch.com/> - FTP-Programm FTS\_PRO und WS\_FTP
- [http://www2.bibl.fh-koeln.de/elektra/ftp\\_hilfe.html](http://www2.bibl.fh-koeln.de/elektra/ftp_hilfe.html) - Shareware-Version von

## FTS\_PRO

- <http://www.jasc.com> - das Grafikprogramm Paint Shop Pro 7
- <http://www.macromedia.com> - das Grafikprogramm Flash
- <http://java.sun.com> - die Java-Entwicklungsumgebung
- <http://www.borland.com> - die JBuilder-Entwicklungsumgebung zur Java-Programmierung
- <http://www.activestate.com/ActivePerl> - Perl-Interpreter (auch für Windows)
- <http://www.perl.com> - Perl-Interpreter
- <http://www.php.net> - PHP4 Win32-Binär und Quellcodedateien
- <http://www.mysql.com> - MySQL-Datenbank

---

❖ Kapitel Inhalt Index **SAMS** ❖ Top Kapitel ❖

---

© [Markt+Technik Verlag](#), ein Imprint der Pearson Education Deutschland GmbH

## Tag Anhang F

Die CD zum Buch

Die Service-CD-ROM, die diesem Buch beiliegt, hat drei Unterverzeichnisse:

- Unter EBOOKS finden Sie dieses Ihnen vorliegende Buch komplett im HTML-Format. So können Sie z.B. eine Lektion auch mal am Laptop durcharbeiten oder auf die Schnelle bereits auf Papier durchgearbeitete Lernschritte noch mal wiederholen. Als Bonusbuch, ebenfalls im HTML-Format, steht dort auch der Bestseller-Titel *JavaScript in 21 Tagen* (ISBN 3-8272-5884-7), der ebenfalls im Markt+Technik-Verlag erschienen ist.
- *TOOLS* heißt das Verzeichnis, das Ihnen eine Reihe von nützlichen Werkzeugen für den Web-Publisher-Alltag zur Verfügung stellt. Unter anderem finden Sie dort Trial-Versionen von Paint Shop Pro 7 und Macromedia Flash 5, Winzip, diverse Spezifikationen und Standards zu HTML, XHTML, CSS1, CSS2, sowie Win32-Binaries und -Sources für PHP4 und Apache Webserver. Bitte beachten Sie die Angaben der einzelnen Anbieter.
- Alle Daten, Grafiken und Skripte für die Beispiele aus den einzelnen Lektionen sind in der Datei *Wprogs.zip* im Unterverzeichnis *Sourcen* zusammengefasst.