HTML II

Was wir nicht machen werden

CGI / Perl

Das Common Gateway Interface (etwa: Allgemeine Vermittlungs-Schnittstelle) ist eine Möglichkeit, Programme im WWW bereitzustellen, die von HTML-Dateien aufgerufen werden können und die selbst HTML-Codes erzeugen sowie an einen Browser senden können.
Wer etwas im Netz bestellt, sich in ein Gästebuch einträgt oder einen Zugriffszähler benutzt, greift auf CGI oder eine vergleichbare Schnittstelle dahinter zurück.
CGI sind Programme, die i.d.R. auf einem Server liegen und bei Aufruf bestimmt Daten verarbeiten; dies geschieht auf dem Server. Aus Sicht des Nutzers steht die CGI-Schnittstelle als Verzeichnis zur Verfügung; meist mit dem Namen cgi-bin.
Die meisten CGI-Programme sind in der Unix-Shell-Sprache oder in Perl geschrieben; während jeder Unix-Rechner Unix-Shell interpretieren kann, muß für Perl ein entspr. Interpreter installiert sein.
Für Anbieter von WWW-Seiten, die sich mit CGI nicht auskennen oder kein CGI zur Verfügung haben, gibt es zahlreiche öffentliche CGI-Dienste, bei denen man sich gratis oder gegen eine geringe Gebühr registrieren lassen kann. Dafür erhält man dann einen HTML-Code, den man in die eigenen Dateien einbaut und der ein entspr. CGI-Programm auf dem Anbieter-Server aufruft.

Ein Beispiel (HTML mit CGI-Script) sieht etwa so aus (Datei »CGI Kommentar«):

```
<HTML>
<HEAD>
<TITLE> Kommentarseite </TITLE>
</HEAD>
<BODY>
<H1> Ihr Kommentar </H1>
<FORM ACTION=„/cgi-bin/comments.pl“ METHOD=post>
Name <INPUT Size=40 MAXLENGTH=40 NAME=„Anwendername“><P>
Text: <TEXTAREA ROWS=5 COLS=70 NAME=„Kommentartext“
WRAP=virtual></TEXTAREA><P>
<INPUT TYPE=„submit“ VALUE=„Absenden“>
</FORM>
</BODY>
</HTML>
```

In der HTML-Datei wird ein Formular mit zwei Eingabefeldern definiert. Das erste Feld (<INPUT SIZE=40…>) ist ein einzeiliges Eingabefeld für den Namen des Anwenders. Das zweite Feld (<TEXTAREA…>) ist ein mehrzeiliger Eingabebereich für einen beliebigen

Text. Durch klicken des Absendeknopfes (<INPUT TYPE=>) wird das Formular abgesendet. Was mit den Daten geschehen soll, steht im einleitenden Formular-Tag.

Java

Java ist eine von Sun Microsystems entwickelte, plattformunabhängge Programmiersprache mit spezieller Ausrichtung auf den Einsatz im WWW. Sie lehnt sich an C/C++ an. Netscape hat sie dann modifiziert und in seine Browsers Interpreter implementiert, die zum Standard wurden.
Java-Programme, die im Internet eingesetzt werden, heißen Applets (Application Snippets, Anwendungsschnipsel). Sie können in HTML-Dateien eingebunden werden, so daß Bildschirmausgaben und (Inter-)Aktionen im Fenster des Browsers erfolgen – sofern er, wie die neueren Versionen – einen Interpreter und ggf. entsprechende Plug-ins hat.
Um Applets selber zu erstellen, bedarf es des Java Developer's Kit. Die meisten laden sich Applets aus dem Internet oder kopieren sie sich aus HTML-Quellcodes besuchter Seiten.

ActiveX

ist eine von Microsoft eingeführte Technik für ausführbare Programmcodes auf WWW-Seiten und als Alternative oder Konkurrenz zu Java gedacht. Es ist – daher – jedoch kein Internet-Standard, sondern der Versuch, spezifische Eigenschaft des Windows-Betriebssystems für WWW-Seiten nutzbar zu machen. Die Komponenten, sog. ActiveX-Controls, sind Programme oder Programm-Module, die sich, ähnlich wie Java-Applets, in HTML-Dateien einbinden lassen.
ActiveX wird derzeit nur vom Explorer direkt ausgeführt, doch gibt es für den Navigator/ Communicator ein Plug-in.

VRML

Die Virtual Reality Modeling Language ist eine Ergänzung zu HTML mit dem Ziel, das Surfen zu einem dreidimensionalen virtuellen Erlebnis zu machen. Die Sprache entstand 1994 und ist für das WWW standardisiert. VRML beschreibt nicht primär Texte und Grafikreferenzen, sondern den vektoriellen Aufbau dreidimensionaler, polygoner Grafikobjekte und deren Abhängigkeiten. Für die Anzeige gibt es spezielle Browsers und auch Add-ons für andere.

XML

Die Extensible Markup Language ermöglicht es, auf der Basis der SGML (Standard Generalized Markup Language), auf der auch HTML fußt, eigene, neue Sprachen zu erfinden bzw. zu definieren. Sie wird, vergleichbar zu den CSS bei HTML, durch die XSL (XML Style Language) ergänzt.
Für normales Homepaging ist XML relativ uninteressant (weil auch trotz anderslautender Gerüchte recht kompliziert), doch für professionelle und komplexe Aufgaben die Wahl. Sie ist ausdrücklich nicht als HTML-Nachfolgerin gedacht, sondern als Parallele.

Links zu spezifischen Abschnitten

Sog. named anchors können auch dazu verwendet werden, um Leser zu einem bestimmten Abschnitt in einem Dokument (demselben oder einem anderen) zu führen – statt zum Beginn eines Textes. »Named anchors« deshalb, weil HTML-Namen eingefügt werden.
Zunächst zu solchen Links in anderen Dokumenten; hier von »Dokument A.html« zu einem bestimmten Abschnitt in Dokument »Journalistenfortbildung.html«. Im Dokument A wird an der gewünschten Stelle ein Anker gesetzt:

```
Das Geb&auml:de des Berliner Verlags ist auch Sitz der
<A HREF=„Journalistenfortbildung.html#BJS“>Berliner Journalisten-
Schule</A>
```

Das Summenzeichen (#) übernimmt die Funktion einer Tabellatormarke im Dokument »Dokument A.html«. Dieses Tab sagt dem Browser, was oben im Fenster als erste Zeile stehen soll, wenn dieser Link aktiviert wird. Im Beispiel die Berliner Journalisten-Schule«. Nun muß im Dokument »Journalistenfortbildung.html« der named anchor »BJS« an die entsprechende Stelle, hier eine Überschrift, gesetzt werden, also:

```
<H2><A NAME=„BJS“>Berliner Journalisten-Schule</A></H2>
```

Es ist zu beachten, daß
(a) in jedem Dokument ein Name nur einmal vergeben werden darf!
(b) solche Links in Dokumenten anderer Autoren nur angelegt werden können, wenn das »Zieldokument« bereits mit anderen named anchors versehen ist und/oder man zur Schreibzugang zum »code source« des Dokuments hat.

Bei solchen Links im selben Dokument (siehe Beispiel »Langtext.html«) ist die Vorgehensweise die gleiche, nur wird der Dokumentenname ausgelassen. Eine Verbindung zum Anker »BJS« (sie wird wegen des Charakters der Funktion auch »Ansprungstelle« genannt) im Dokument »Journalistenfortbildung.html« sähe dann so aus:

```
…weitere Informationen zur <A HREF=„#BJS“>Berliner Journalisten-
Schule</A> finden sich an anderer Stelle in diesem Dokument.
```

Dabei darf natürlich nicht vergessen werden, den Tag <A NAME=> an die Stelle im Dokument zu setzen, an die der Link springen soll (vgl. weiter oben)!

E-Mail-Verweise

Sie erinnern sich an die Möglichkeit, einen E-Mail-Verweis in das HTML-Dokument einzubauen:

```
<A HREF=„mailto:emailinfo@host“>Ankerwort</A>
```

Es läßt sich auch ein Verweis definieren, mit dem der Anwender zwei oder mehr Empfänger erreicht:

```
<A HREF=„mailto:egon@a-host?cc=albert@b-host;erna@c-host“>Anker</A>
```

Nach der ersten E-Mail-Adresse steht ein »?«, danach folgt ein »cc« (cover copy) für die folgende(n) Adresse(n), die jeweils durch ein »;« getrennt werden.

Um es dem Anwender noch einfacher zu machen, läßt sich auch das »subject« der E-Mail vorgeben:

```
<A HREF=„mailto:anna@x-host?subject=feedback“>Anker</A>
```

Und für die, die nicht schreiben können, wird auch gleich der Text im »body« der E-Mail vorformuliert:

```
<A HREF=„mailto:hirni@y-host?body=Mein feedback:%0Gut so!“>Anker</A>
```

Die Zeichenfolge »%« und Null erwirkt einen Zeilenumbruch.

Diese vordefinierten Angaben lassen sich auch kombinieren, wobei zwischen ihnen jeweils ein »&« steht:

```
<A HREF= „mailto:erwin@z-host?cc=rolf@m-host&subject=Hallo&body=feed
back“>Text</A>
```

Mehrspaltiger Text

Beim Navigator ab 3.0 gibt es eine einfachere Methode als die der »blinden« Tabelle, Text im Stil einer Zeitung oder Zeitschrift mehrspaltig darzustellen. Leider ist dies noch kein HTML-Standard. Außerdem gilt es zu berücksichtigen, daß Bilder, Java-Applets u.a. sich nicht an die vorgegebene Spaltenbreite halten können, so daß sie, wenn sie breiter als diese sind, in die Nachbarspalte ragen.
Der mehrspaltig zu setzende Text steht im »Container« der Tags

```
<MULTICOL></MULTICOL>
```

Zwingend erforderlich ist die Angabe der Spaltenzahl mit dem Attribut COLS=n
Mit dem Attribut WIDTH=n(%) wird die Gesamtbreite der Spalten beeinflußt; also entweder in Pixelwerten oder in % der Fensterbreite. Mit dem Attribut GUTTER=n wird die Breite des Zwischenschlags (default ist 10 Pixel) definiert.

Tabellen

Vordefinieren

Die Darstellung einer Tabelle ergibt sich zwar automatisch aus den definierten Zeilen und Spalten. Doch für einen WWW-Browser ist es nicht ganz einfach, die Darstellung frühzeitig zu ermitteln. Er muß erst die gesamte Tabelle einlesen, bevor er irgendetwas davon darstellen kann. Bei großen Tabellen kann dies zu unschönen leeren Bildschirmen während des Seitenaufbaus führen.

HTML 4.0 bietet eine neue Syntax an, um dem Browser gleich zu Beginn der Tabelle mitzuteilen, wie viele Spalten die Tabelle hat. Dadurch kann der Browser die Tabelle schneller aufbauen, d.h. bereits Teile der Tabelle anzeigen, bevor die gesamte Tabelle eingelesen ist. Etliche Browser (Netscape 3.x, 4.x, Explorer 3.x) interpretieren diese Angaben noch nicht (Datei »Tabellen.html«).

Beispiel 1:

```
<table border>
    <colgroup>
      <col width=80>
      <col width=100>
      <col width=320>
    </colgroup>
   <tr>
     <td>1. Zeile, 1. Spalte</td>
     <td>1. Zeile, 2. Spalte</td>
     <td>1. Zeile, 3. Spalte</td>
   </tr>
      <!-- usw. andere Zeilen der Tabelle -->
</table>
```

Beispiel 2:

```
<table border>
    <colgroup width=200 span=3>
    </colgroup>
   <tr>
     <td>1. Zeile, 1. Spalte</td>
     <td>1. Zeile, 2. Spalte</td>
     <td>1. Zeile, 3. Spalte</td>
   </tr>
      <!-- usw. andere Zeilen der Tabelle -->
</table>
```

Beispiel 3:

```
<table border>
    <colgroup>
      <col width="4*">
      <col width="2*">
      <col width="1*">
    </colgroup>
   <tr>
     <td>1. Zeile, 1. Spalte</td>
     <td>1. Zeile, 2. Spalte</td>
     <td>1. Zeile, 3. Spalte</td>
   </tr>
      <!-- usw. andere Zeilen der Tabelle -->
</table>
```

Mit <colgroup> leitet man hinter dem einleitenden <table>-Tag eine Vorab-Definition der Tabellenspalten ein. Dabei gibt es zwei Möglichkeiten: entweder man möchte unterschiedlich

breite Tabellenspalten haben. Dann geht man so vor wie im Beispiel 1. Oder man hat eine Tabelle, in der alle Spalten die gleiche einheitliche Breite haben sollen. Dann kann man vorgehen wie im Beispiel 2.

Im Beispiel 1 enthält das <colgroup>-Tag keine weiteren Angaben. Dafür notiert man unterhalb dieses Tags für jede einzelne gewünschte Tabellenspalte je einen Befehl <col>. Das erste col-Tag definiert die erste Spalte, das zweite die zweite Spalte usw. Wird keine weitere Angabe gemacht, wird die Breite der Spalten automatisch aufgrund des Tabelleninhalts ermittelt.
Mit width=n / n% läßt sich jedoch eine Spaltenbreite für die einzelnen Spalten vorgeben. Width=100 erzwingt beispielsweise eine Spaltenbreite von 100 Pixeln, width=33% eine Breite von einem Drittel des Anzeigefensters.

Im Beispiel 2 werden keine Befehle vom Typ <col> notiert. Stattdessen notiert man im einleitenden <colgroup>-Tag das Attribut span=n. Man gibt hinter dem Istgleichzeichen die gewünschte Anzahl der Spalten an. Mit dem Attribut width=n kann in diesem Fall eine einheitliche Spaltenbreite für alle Spalten definiert werden.

Bei der Angabe width=n besteht neben der Möglichkeit, Pixel oder Prozentwerte anzugeben, auch noch eine dritte Möglichkeit:
Es läßt sich das relative Breitenverhältnis der Spalten untereinander bestimmen, unabhängig davon, wie breit die Tabelle im Verhältnis zum Anzeigefenster ist. Eine solche Möglichkeit stellt das Beispiel 3 vor. Bei Breitenangaben dieser Art notiert man hinter width= eine Zahl und dahinter ein Sternzeichen. Das Sternzeichen ist dabei nur ein Signalzeichen für den Browser, daß er die Zahlen davor nicht als Pixel interpretieren soll. Wichtig sind die Zahlen. Im Beispiel 3 werden drei Spalten definiert, bei denen die relativen Zahlen 4, 2 und 1 in der Summe 7 ergeben. Damit ist eine Tabelle definiert, bei der die erste Spalte vier Siebtel der Tabellenbreite einnimmt, die zweite Spalte zwei Siebtel, die dritte Spalte ein Siebtel. So richtig zur Geltung kommt dieses relative Spaltenverhältnis aber erst, wenn außerdem eine Breite für die gesamte
Tabelle angeben ist.

Außenrahmen

Es läßt sich bestimmen, welche Seiten eines Außenrahmens angezeigt werden und welche nicht.

Beispiel:

```
<TABLE BORDER FRAME=box>
    <!--Inhalt der Tabelle-->
</TABLE>
```

Voraussetzung für diese Angaben ist das Attribut »border« oder »border=n« im einleitenden »table«-Tag. Dadurch wird ein Außenrahmen angezeigt.

Mit dem Attribut »frame« kann man dann bestimmen, an welchen Seiten der Tabellenrahmen gezogen werden soll.

Mit der Angabe »FRAME=box« – so wie im Beispiel – erreicht man, daß der Tabellenrahmen oben, links, rechts und unten sichtbar dargestellt wird (die Angabe ist identisch mit dem, was die Angabe »BORDER« bewirkt – auch »FRAME=border« ist erlaubt und bewirkt das Gleiche).

Folgende andere Angaben sind möglich:

Mit frame=void wird überhaupt keine Tabellenrahmen angezeigt.
Wird »border« angeben, werden jedoch die Gitternetzlinien der Tabelle sichtbar angezeigt. Die
Tabelle sieht dann also aus wie ein an allen Seiten offenes Gitter.
Mit frame=above wird nur am oberen Rand der Tabelle eine Rahmenlinie angezeigt.
Mit frame=below wird nur am unteren Rand der Tabelle eine Rahmenlinie angezeigt.
Mit frame=hsides (horizontal sides) wird nur am oberen und am unteren Rand der Tabelle eine
Rahmenlinie angezeigt.
Mit frame=vsides (vertical sides) wird nur am linken und am rechten Rand der Tabelle eine Rahmenlinie angezeigt.
Mit frame=lhs (left hand side) wird nur am linken Rand der Tabelle eine Rahmenlinie angezeigt.
Mit frame=rhs (right hand side) wird nur am rechten Rand der Tabelle eine Rahmenlinie angezeigt.

Netscape 4.x interpretiert diese Angaben noch nicht.

Horizontale und vertikale Abstände

Wiederum nur beim Navigator ab 3.0 gibt es die Möglichkeit, horizontale und vertikale Leerräume im Text zu bestimmen. Sie können in Zusammenhang mit dem <P> bei horizontalen (Einrückungen) oder auch an beliebiger Stelle bei vertikalen (Abstände) Räumen gesetzt werden.

Der entsprechende Tag ist der SPACER, mit dem Attribut TYPE=HORIZONTAL/ VERTICAL wird die Ebene angezeigt, mit SIZE=n die gewünschte Größe in Pixeln.
Es lassen sich auch mehrere Zeilen horizontal stufenartig einrücken, so daß ein Treppeneffekt entsteht. Und da solche Einrückungen selbst innerhalb von Zeilen möglich sind, lassen sich selbst mehrspaltige Treppen erzeugen (Dokument »Treppe.html«):

```
<HTML>
<HEAD>
<TITLE> Treppe.html</TITLE>
<BODY>
Die erste Zeile<SPACER TYPE=horizontal SIZE=50>Mit Abstand<BR>
  <SPACER TYPE=horizontal SIZE=20>Die zweite Zeile<SPACER TYPE=horizontal
SIZE=50>Mit Abstand<BR>
  <SPACER TYPE=horizontal SIZE=40>Die dritte Zeile<SPACER TYPE=horizontal
SIZE=50>Mit Abstand<BR>
  <SPACER TYPE=horizontal SIZE=60>Die vierte Zeile<SPACER TYPE=horizontal
SIZE=50>Mit Abstand<BR>
```

```
  <SPACER TYPE=horizontal SIZE=40>Die fünfte Zeile<SPACER TYPE=horizontal
SIZE=50>Mit Abstand<BR>
  <SPACER TYPE=horizontal SIZE=20>Die sechste Zeile<SPACER TYPE=horizontal
SIZE=50>Mit Abstand<BR>
  Die siebte Zeile<SPACER TYPE=horizontal SIZE=50>Mit Abstand<BR>
</BODY>
</HTML>
```

Mit dem »Spacer« lassen sich sogar »unsichtbare Bilder« konstruieren. Diese, mit frei wählbarer Größe, werden vom umgebenden Text umflossen wie referenzierte Grafiken. Die Syntax hierfür lautet (Datei: »Block.html«):

```
<SPACER TYPE=block WIDTH=n HEIGHT=n ALIGN=left/right>
```

Mit dem »block« wird ein Rechteck mit definierter Höhe und Breite in Pixeln erzeugt. Bei der Ausrichtung des Blocks umfließt der Text bei »left« ihn rechts und bei »right« links. Folgt dieser Befehl gleich nach dem BODY-Tag und ist die »height« ausreichend groß gewählt, wird ein der »width« entsprechender Seitenrand erzeugt.

Externe Bilder, Töne und Animationen

Manchmal möchte man, daß ein Bild als separates Dokument geöffnet wird, wenn der Betrachter einen Link an einem Wort oder einer kleineren Inline-Version im Dokument herstellt. Solch ein externes Bild ist dann nützlich, wenn das Laden des Dokuments nicht durch größere Darstellungen beeinträchtigt werden soll. Als Tag zu einem externen Bild gilt:

```
<A HREF=„Bildname.gif“>Ankerpunktbezeichnung</A>
```

Wird ein kleines Bild als Link zu seiner größeren Version definiert, geht es mit:

```
<A HREF=„Großes Bild.jpeg“><IMG SRC=„Kleines Bild.jpeg“></A>
```

Der Betrachter sieht das kleine Bild und klickt es an, um das größere JPEG-Bild zu sehen.

Dieselbe Syntax gilt für Links zu externen Tönen und Animationen; es differiert hier lediglich der Suffix des zu verbindenden Dokuments. So stellt etwa

```
<A HREF=„GeburtsTag.mov“>Ankerpunktbezeichnung</A>
```

einen Link zu einem QuickTime-Film dar.

Hintergrundmusik

Es läßt sich bestimmen, daß beim Aufruf einer HTML-Datei Hintergrundmusik ertönt. Leider sind die Lösungen beim Navigator und beim Explorer proprietär, d.h. folgen (wenn sie es denn tun!) unterschiedlichen Anweisungen.

Bei Microsoft lautet die Syntax:

```
<HEAD>
<BGSOUND SRC=„datei.mid“ LOOP=infinite>
</HEAD>
```

Und bei Netscape:

```
<BODY>
<EMBED SRC=„datei.mid“ AUTOSTART=true LOOP=true HIDDEN=true HEIGHT=0
WIDTH=0>
```

BGSOUND im Kopfbereich bzw. EMBED im Body definiert die Anzeige einer Musikdatei, die als SOURCE angegeben wurde. Es sollte sich dabei möglichst um die Typen .mid, .wav oder .au handeln.
Um die Anzeige des Players zu unterdrücken, erfordert der Navigator die Angaben zu Breite, Höhe und versteckt. Auch der automatische Start muß hier explizit angegeben werden. Ferner läßt sich bestimmen, ob die Musik einmal, mehrmals oder endlos (bis zum Aufruf einer neuen Datei) abgespielt wird. Beim Explorer wird der »loop« mit der Zahl der Wiederholungen oder mit endlos (infinite) markiert. Beim Navigator gibt es nur endlos («loop=true») oder einmal («loop» weglassen).

Meta-Tags und andere »Head«-Einträge

Diese Tags sind grundsätzlich im HEAD-Bereich zu setzen! Daher sind sie für den Betrachter im Browserfenster nicht sichtbar. Sie dienen auch nicht in erster Linie zu dessen Information, sondern dem »HTTP-Protokoll-Verkehr«, d.h. der Information des Browsers, von Servern, der Suchmaschinen oder der Verknüpfung von Dokumenten.

Zielfensterbasis

Diese Angabe ist natürlich nur bei der Verwendung von Frames sinnvoll. Mit ihr wird festgelegt, daß alle Verweise einer Datei, die in einem Frame angelegt wird, in einem best. anderen Frame angezeigt werden (solange kein anderer Frame angegeben wird). Da viele Verweise in einer Datei in einem best. anderen Frame angezeigt werden sollen, erspart dies etliche Tipparbeit und verkleinert die Datei.
Beispiel:

```
<HEAD>
<BASE TARGET=„Fenster-links“>
… …
</HEAD>
```

Meta-Elemente

für Suchprogramme sind z.B.:

```
<HEAD>
<META NAME=„description“ CONTENT=„Hier steht ein beschreibender
Text/abstract“>
<META NAME=„author“ CONTENT=„mein Name“>
<META NAME=„keywords“ CONTENT=„entspr. Stichwörter“>
<META NAME=„date“ CONTENT=„1999-08-12“>
… …
</HEAD>
```

Deutsche Umlaute und Sonderzeichen sollten vermieden oder »maskiert« werden.

Bei internationaler Orientierung lassen sich Stichwörter auch nach Sprachen organisieren:

```
<META NAME=„keywords“ LANG=„de“
        CONTENT=„Ferien, Zypern, Sonnenschein“>
<META NAME=„keywords“ LANG=„en“
        CONTENT=„holiday, Cyprus, sunshine“>
<META NAME=„keywords“ LANG=„en-us“
        CONTENT=„vacation, Cyprus, sunshine“>
<META NAME=„keywords“ LANG=„fr“
CONTENT=„vacances, Chypre, soleil“>
```

Will man ausnahmsweise einem Suchprogramm verbieten, Inhalte aus der HTML-Datei an seine Datenbank zu übermitteln, setzt man

```
<META NAME=„robots“ CONTENT=„noindex/none“>
```

Und hier noch eine Einladung an Suchprogramme:

```
<META NAME=„revisit-after“ CONTENT=„30 days“>
```

Das heißt: Such-Robot, bitte nach 30 Tagen wieder vorbeikommen!

Derartige Meta-Informationen zum Inhalt von HTML-Dateien sind insbesondere dann sinnvoll bis nötig, wenn wichtige Dateibestandteile wie etwa die Einstiegsseite (fast) völlig aus Grafikelementen bzw. –referenzen besteht.

Meta-Tags für Browsers sind u.a.
Angaben zur Default-Sprache:

Mit diesen Tags läßt sich bestimmen, welcher Zeichensatz in der HTML-Datei vorliegt und welche Ergänzungssprachen für Scripts und Style-sheets verwendet werden. Etwa:

```
<META HTTP-EQUIV=„content-type“ CONTENT=„text/html; charset=iso-8859-1“>
```

Will sagen: Es geht um einen Zeichensatz (»type«) für das HTTP, den sog. MIME-Type (Multipurpose Internet Mail Extension, Methode zur Differenzierung von Dateiformaten), der in HTML immer »text/html« heißt, und den Zeichensatz iso-8859-1 für westeuropäische Sprachen (abgesetzt durch ein Semikolon!).

<META HTTP-EQUIV=„Content-Script-Type“ CONTENT=„text/javascript“>

Hier wird die Scriptsprache festgelegt; in diesem Falle mit dem MIME-Type »text/javescript«.

<META HTTP-EQUIV=„Content-Style-Type“ CONTENT=„text/css“>

Und hier dann die Bestimmung der Stylesprache als Cascading Style Sheet.
Zu beachten ist die unterschiedliche Groß-klein-Schreibung beim »equiv«.

Bei Websites mit häufig wechselndem oder ergänztem Inhalt kann ein Browser gezwungen werden, die Daten nicht aus dem Browser-Cache oder dem Proxy-Cache zu holen, sondern vom Original-Server:

<META HTTP-EQUIV=„expires“ CONTENT=„0“>
oder
<META HTTP-EQUIV=„expires“ CONTENT=„Tue, 20 Aug 1999 15:25:30 GMT“>

Im ersten Tag wird dem Browser befohlen, nach Null Sekunden eine Datei im Cache wieder vom Originalserver abzurufen (bei 43200 also nach 12 Stunden), im zweiten Tag ist dieser Zeitpunkt mit einem konkreten Datum angegeben.

Browserinterne Ressourcen

Auch in den Browsers sind intern Bilddateien implementiert. Man denke nur an die Darstellung im nicht grafikorientierten FTP-Protokoll oder an die Anzeigen von »bad links« bei nicht darstellbaren Grafiken. Dies funktioniert jedoch – bislang – nur beim Netscape und, mit z.T. anderen SRC-Namen, dem IBM Web Explorer.
Drei Beispiele sind (Datei: »Ressourcen intern.html«):

<IMG SRC=„internal-gopher-index“>
und
<IMG SRC=„internal-news-next-thread-gray“>
sowie
<IMG SRC=„internal-icon-notfound“>

Grafiken als Links

Anstelle eines Verweistextes/Links kann auch eine beliebige Grafik definiert werden. Dann ist sie anklickbar und führt den gewünschten Link aus:

<A HREF=„Datei.html“><IMG SRC=„Bild.gif“></A>

Im Browserfenster ist diese Grafik dann umrahmt und somit als Link erkennbar; zudem ändert sich die Cursorfigur zum »Zeigefinger«, wenn sie überlaufen wird. Der Rahmen kann durch das Attribut »BORDER=0« im IMG SRC-Tag unterdrückt werden.

Image Maps

Dies sind verweis-sensitive Grafiken, in denen ein oder mehrere definierte Detail/s als Link/s funktioniert/funtionieren. Die Syntax sieht wie folgt aus (Datei: »ImageMap/ImageMap.html«):

```
<MAP NAME=„image-map-test“>
<AREA SHAPE=rect COORDS=„x1,y1,x2,y2“ HREF=„Datei1.html“>
<AREA SHAPE=circle COORDS=„x,y,r“ HREF=„Datei2.html“>
<AREA SHAPE=polygon COORDS=„x1,y1,x2,y2…xn,yn“ HREF=„Datei3.html“>>
</MAP>
```

Im »MAP«-Container werden die verweis-sensitiven Flächen definiert. Der vergebene Name muß nicht mit dem (Datei-)Namen der Grafik identisch sein; er ist lediglich ein »Anker«-Name. Er sollte nicht zu lang sein und darf keine Leerzeichen und Umlaute enthalten, als Sonderzeichen nur den Unterstrich »_«.

Die sensitiven Flächen werden mit AREA definiert. Mit SHAPE werden Rechtecke (»rect«), Kreise (»circle«) oder Vielecke (»polygon«) ausgewiesen. Natürlich lassen sich in einer entspr. großen Grafik mehrere Flächen, auch gemischt, angeben.

Die COORDS-Werte bezeichnen die Koordinaten der sensitiven Fläche als absolute Pixel-Werte innerhalb der Grafik, die durch Kommata ohne Leerzeichen getrennt werden.

Die Koordinaten bedeuten beim Viereck:
x1 linke obere Ecke, Pixel von links
y1 linke obere Ecke, Pixel von oben
x2 rechte untere Ecke, Pixel von links
y2 rechte untere Ecke, Pixel von oben

Ein Kreis wird so definiert:
x Mittelpunkt, Pixel von links
y Mittelpunkt, Pixel von oben
r Radius in Pixel

Bei einem Vieleck gilt:
x Pixel einer Ecke von links
y Pixel einer Ecke von oben
Die Zahl der Ecken ist theoretisch nicht begrenzt. Das Polygon wird geschlossen, wenn man sich eine Gerade von der letzten zur ersten definierten Ecke vorstellt.

Mit HREF wird, wie bei Links üblich, die referenzierte aufzurufende Datei angegeben.

Die Anweisung MAP kann an beliebiger Stelle innerhalb des Bodys einer HTML-Datei stehen, denn sie wird noch nicht exekutiert. Dies geschieht erst wenn und dort, wo die gemeinte Grafik plaziert wird. Das geschieht herkömmlich mit
<IMG SRC=„grafik.gif“ USEMAP=„#image-map-test“ BORDER=0>

Daher kann das MAP-Notat durchaus in einer anderen HTML-Datei stehen als in der, in der die Image Map positioniert ist. Im USEMAP-Attribut muß dann natürlich der korrekte Pfad angegeben werden!

Formulare/Eingabefelder

Sie können an einer beliebigen Stelle innerhalb des Dateikörpers einer HTML-Datei ein Formular nach zwei versch. Mustern definieren.

Beispiel 1:

```
<FORM ACTION="mailto:muenz@csi.com" METHOD=POST enctype="text/plain">
... Elemente des Formulars wie Eingabefelder, Auswahllisten, Buttons usw. ...
</FORM>
```

Beispiel 2:

```
<FORM ACTION="/cgi-bin/auswert.pl" METHOD=GET>
... Elemente des Formulars wie Eingabefelder, Auswahllisten, Buttons usw. ...
</FORM>
```

Mit <form> definiert man ein Formular. Alles, was zwischen diesem einleitenden Tag und dem abschließenden Tag </form> steht, gehört zum Formular. Das sind hauptsächlich Elemente des Formulars wie Eingabefelder, Auswahllisten oder Buttons. Um die Elemente zu plazieren, braucht man jedoch auch Textabsätze (Absatzschaltungen) und erzwungene Zeilenumbrüche. Darüber hinaus läßt sich zwischen <form> und </form> auch Text eingeben und diesen Text wie üblich mit HTML-Befehlen formatieren. Auch Grafiken, Verweise, Tabellen, Multimedia-Elemente sind mitten im Formular erlaubt. So kann man das Formular optisch aufwerten und mit erklärendem Text usw. versehen.

Im einleitenden <form>-Tag gibt man mit action= an, was mit den ausgefüllten Formulardaten passieren soll, wenn der Anwender das Formular abschickt. Die Angabe sollte in Anführungszeichen stehen.

Die Angabe bei action= ist entweder eine E-Mail-Adresse (normalerweise die eigene) mit vorangestelltem mailto:. Dann werden die ausgefüllten Formulardaten an diese E-Mail-Adresse geschickt.
Oder man ruft ein Programm auf dem Server-Rechner, meist ein CGI-Programm, auf, das die Daten weiterverarbeitet - wieim zweiten Beispiel. Bei der Adressierung des CGI-Programms gelten die gleichen Regeln wie bei Verweisen.

Bei der Formulardefinition muß als nächstes die Übertragungsmethode angegeben werden. Dabei gibt es zwei Möglichkeiten:

Wird »method=get« gewählt, werden die Daten des ausgefüllten Formulars auf WWW-Servern mit installiertem HTTP-Protokoll in der CGI-Umgebungsvariablen QUERY_STRING gespeichert. Das CGI-Programm muß den Inhalt dieser Umgebungsvariablen auslesen und verarbeiten.

Wird »method=post« gewählt, werden die Daten des augefüllten Formulars auf dem Server-Rechner von »stdin« zur Verfügung gestellt, und das CGI-Programm muß sie behandeln wie eine Benutzereingabe, die auf der Kommandozeile gemacht wurde. Da in diesem Fall kein EndOfFile-Signal (EOF) gesendet wird, muß das CGI-Programm die CGI-Umgebungsvariable CONTENT_LENGTH auslesen, um die Länge der übermittelten Daten und damit deren Ende zu ermitteln.

Wenn man sich Formulardaten per E-Mail zuschicken läßt, benutzt man immer »method=post«. Außerdem sollte man bei E-Mail-Empfang von Formulardaten immer die Angabe enctype="text/plain" mit angeben. Denn Formulardaten sind normalerweise nach einem bestimmten Schema formatiert, das für auswertende Programme recht gut geeignet ist, aber für Menschen keine Freude zu lesen ist. Mit der genannten Angabe erhält man zumindest von Anwendern, die Ihr Formular mit einem modernen WWW-Browser ausfüllen, ordentlich formatierte E-Mails.

Nicht alle Browser beherrschen das Versenden von Formularen per E-Mail.

Zielfenster bei frames

Wer mit Frames arbeitet und in einem Frame-Fenster ein Formular hat, nach dessen Absenden die Server-Antwort (zum Beispiel die Ausgabe eines CGI-Scripts) in einem anderen Frame-Fenster angezeigt werden soll, kann das Zielfenster für die Server-Antwort angeben:

```
<FORM ACTION="/cgi-bin/auswert.pl" METHOD=POST TARGET="Daten">
  ... Elemente des Formulars wie Eingabefelder, Auswahllisten, Buttons usw. ...
  </FORM>
```

Mit dem Attribut target= kann man im einleitenden <form>-Tag den Namen des Frame-Fensters angeben, in dem die Server-Antwort ausgegeben werden soll. Sowohl Netscape als auch der MS Internet Explorer interpretieren diese Angabe.

Einzeilige Eingabefelder

Einzeilige Eingabefelder dienen zur Aufnahme von einem oder wenigen Wörtern oder einer Zahl (Datei »Formular-Eingabe 1.html«).

Beispiel 1:

```
Ihr Spitzname: <INPUT NAME="Spitzname" SIZE=60 MAXLENGTH=60>
<BR>
Ihr Lieblings-Star: <INPUT TYPE=text NAME="LieblingsStar" SIZE=40
MAXLENGTH=40>
<BR>
Ihre heimliche Leidenschaft: <INPUT NAME="Leidenschaft" SIZE=30
MAXLENGTH=60>
```

<input ... > definiert ein einzeiliges Eingabefeld. Der Vollständigkeit halber kann man die Angabe type=text dazusetzen.

Jedes Eingabefeld muß einen internen Bezeichnernamen erhalten, und zwar mit der Angabe name=. Der Name sollte nicht zu lang sein und darf keine Leerzeichen und keine deutschen Umlaute enthalten. Man benutze als Sonderzeichen höchstens den Unterstrich »_« und setze Sie den Namen in Anführungszeichen.
Ferner sollte bei einzeiligen Eingabefeldern immer die Anzeigelänge in Zeichen mit size= sowie die interne Feldlänge in Zeichen maxlength= bestimmt werden. Beide Angaben bedeuten die Anzahl Zeichen. Wenn die interne Feldlänge maxlength größer ist als die angezeigte Feldlänge size (wie im dritten der Beispiele), dann wird bei längeren Eingaben automatisch gescrollt (im Beispiel also ab dem 41. eingegebenen Zeichen).

Beispiel 2:

```
<FORM>
<TABLE>
<TR>
<TD ALIGN=“right“>Ihr Vorname:
<TD><INPUT TYPE=“text“ SIZE=40 MAXLENGTH=40>
</TD>
</TR>
<TR>
<TD ALIGN=right>Ihr Zuname:
<TD><INPUT TYPE=“text“ SIZE=40 MAXLENGTH=40>
</TD>
</TR>
</TABLE>
</FORM>
```

Das Beispiel zeigt, wie mit Hilfe einer blinden Tabelle Beschriftung und Eingabefelder eines Formulars ordentlich formatiert werden kann.

Man kann ein einzeiliges Eingabefeld mit einem Inhalt vorbelegen.

```
Ihre Lieblings-URL:
<INPUT NAME="LieblingsURL" MAXLENGTH=40 SIZE=40
VALUE="http://www.spiegel.de">
```

Eingabefelder mit vorbelegtem Inhalt werden wie gewöhnliche Eingabefelder definiert. Mit dem zusätzlichen Attribut value= kann man einen Text angeben, mit dem das Feld vorbelegt wird. Der Text muß in Anführungszeichen stehen.

Passwörter

Für die Eingabe von Geheimnummern, Passwörtern usw. gibt es einen speziellen Typ von Eingabefeld. Die eingegebenen Zeichen werden dabei durch Platzhalter (meistens Sternchen oder Bullets) dargestellt, so daß Personen im Raum des Anwenders nicht zufällig das eingegebene Passwort mitlesen können.

```
Ihre Seriennummer: <INPUT TYPE=password NAME="Zugangsnummer"
MAXLENGTH=10 SIZE=10>
```

Eingabefelder für Passwörter werden wie gewöhnliche Eingabefelder definiert. Mit der zusätzlichen Angabe »type=password« wird bestimmt, daß es sich um ein Passwort-Feld handelt.

Passwörter werden trotz der verdeckten Eingabe im Klartext über das Internet übertragen. Weisen Sie Anwender in ernsthaften Zusammenhängen auf diese Tatsache bitte explizit hin. Ein Paßwortfeld in HTML allein ist noch keine Abfrage für geschützte WWW-Seiten. Bitte kontaktieren Sie beim Wunsch, paßwortgeschützte WWW-Seiten anzubieten, Ihren Provider.

Mehrzeilige Eingabefelder

Diese dienen der Aufnahme von Kommentaren, Nachrichten usw. (Datei »Formular-Eingabe 2.html«)

Beispiel:

```
Was halten Sie davon, da&szlig; HTML-Dozenten so wenig verdienen:
</p>
<TEXTAREA NAME="HTML_Dozenten" ROWS=10 COLS=50></TEXTAREA>
```

<textarea> leitet ein mehrzeiliges Eingabefeld ein. Jedes mehrzeilige Eingabefeld muß einen internen Bezeichnernamen erhalten, und zwar mit der Angabe name=. Der Name sollte nicht zu lang sein und darf keine Leerzeichen und keine deutschen Umlaute enthalten. Man verwende als Sonderzeichen höchstens den Unterstrich »_» und setze den Namen in Anführungszeichen.
Dahinter folgen zwei Angaben zur Anzeigegröße des Textfelds. rows= bestimmt die Anzahl der angezeigten Zeilen, cols= die Anzahl der angezeigten Spalten. »Spalten« bedeutet dabei die Anzahl Zeichen (pro Zeile).
Mit </textarea> schließt man das mehrzeilige Eingabefeld ab. Das End-Tag ist nötig und darf nicht weggelassen werden.

Die Angaben rows= und cols= bestimmen lediglich die Anzeigegröße des Eingabebereichs, nicht die Länge des erlaubten Textes. Die ist theoretisch unbegrenzt. WWW-Browser statten die mehrzeiligen Eingabefelder bei der Anzeige üblicherweise mit Scrollbalken aus, so daß der Anwender bei längeren Eingaben scrollen kann.

Das <textarea>-Tag erzeugt einen eigenen Absatz.

Textvorbelegung bei mehrzeiligen Eingabefeldern

Hiermit wird ein mehrzeiliges Eingabefeld mit Inhalt vorbelegt.

Beispiel:

```
Was halten Sie von der Vermicrosoftung des Internet:
<P>
<TEXTAREA NAME="Vermicrosoftung" ROWS=10 COLS=50>
Schreiben Sie sich Ihren Frust von der Seele!
</TEXTAREA>
```

Mehrzeilige Eingabefelder mit vorbelegtem Text werden wie gewöhnliche mehrzeilige Eingabefelder definiert. Die Textvorbelegung definieren Sie nach dem einleitenden <textarea>-Tag und vor dem abschließenden </textarea>.

Umbruch bei mehrzeiligen Eingabefeldern kontrollieren

Normalerweise erfolgt innerhalb mehrzeiliger Eingabefelder kein automatischer Zeilenumbruch, was auf viele Anwender, die moderne Eingabeformen kennen, irritierend wirkt. Es gibt jedoch eine Möglichkeit, einen automatischen Zeilenumbruch zu erzwingen.

Beispiel 1:

```
Was halten Sie vom Schlagwort &amp;quot;rechtsfreier Raum Internet&amp;quot;:
<p>
<textarea name="RaumInternet" rows=10 cols=50 WRAP="virtual">
</textarea>
```

Beispiel 2:

```
Was halten Sie vom Gerede um Kinderpornos und Neonazis im Internet:
<P>
<TEXTAREA NAME="Gerede" ROWS=10 COLS=50 WRAP="physical">
</TEXTAREA>
```

Mehrzeilige Eingabefelder mit Umbruchkontrolle werden wie gewöhnliche mehrzeilige Eingabefelder definiert. Mit dem Attribut wrap= kann man den Zeilenumbruch steuern (wrap = Umbruch). Mit »wrap=virtual« bewirkt man, daß der Text bei der Eingabe automatisch umbrochen wird (virtual = scheinbar). Beim Absenden des Formulars werden jedoch keine Zeilenumbruchzeichen übertragen. Mit »wrap=physical« erreicht man ebenfalls, daß der Text bei der Eingabe automatisch umbrochen wird (physical = tatsächlich). Die Zeilenumbruch-zeichen werden beim Absenden des Formulars jedoch mit übertragen. Mit »wrap=off« stellt man den Zeilenumbruch aus (Voreinstellung).

Formulare/Auswahllisten

Definieren

Man kann dem Anwender eine Liste mit Einträgen anbieten, aus der er wählen kann. Der Text des gewählten Eintrags wird beim Abschicken des Formulars übertragen (Datei »Auswahllisten.html«):

```
Ihr Favorit:
<SELECT NAME="top5" SIZE=3>
<P>
<OPTION> Heino
<OPTION> Bob Dylan
<OPTION> Roy Black
<OPTION> Roberto Blanco
<OPTION> Marianne Rosenberg
</SELECT>
```

<select> leitet eine Auswahlliste ein. Jede Auswahlliste muß einen internen Bezeichnernamen erhalten, und zwar mit der Angabe name=. Der Name sollte nicht zu lang sein und darf keine Leerzeichen und keine deutschen Umlaute enthalten. Man erwende als Sonderzeichen höchstens den Unterstrich »_« und setze den Namen in Anführungszeichen.
Mit dem Attribut size= bestimmt man die Anzeigegröße der Liste, d.h. wie viele Einträge angezeigt werden sollen. Wenn die Liste mehr Einträge enthält als angezeigt werden, kann der Anwender in der Liste scrollen. Wird »size=1« angeben, definiert man eine »Drop-Down-Liste«.

<option> definiert jeweils einen Eintrag der Auswahlliste. Hinter <option> muß der Text des Listeneintrags stehen. Man kann so viele Listeneinträge definieren wie man will.
Ein Abschluß-Tag </option> ist erlaubt, aber nicht zwingend erforderlich.
Mit </select> wird die Auswahlliste abgeschlossen.
Die Breite der Listenanzeige wird automatisch ermittelt, abhängig vom längsten Eintrag.

Auswahlliste mit Mehrfachauswahl

Wird nichts anderes angegeben, kann der Anwender aus einer Auswahlliste genau einen Eintrag auswählen. Man kann auch eine Mehrfachauswahl erlauben.

```
Alle Ihre Favoriten aus der 70er-Jahre-Liste:
</P>
<SELECT NAME="top70s" SIZE=5 MULTIPLE>
<OPTION> The Beatles
<OPTION> The Rolling Stones
<OPTION> Dave Clark 5
<OPTION> The Beach Boys
<OPTION> Die Egerl&auml;nder
</SELECT>
```

Auswahllisten mit Mehrfachauswahl werden wie gewöhnliche Auswahllisten definiert. Die Mehrfachauswahl erlaubt man durch das zusätzliche Attribut »multiple« im einleitenden <select>-Tag.

Eine Mehrfachauswahl ist für Anwender nicht unmittelbar erkennbar. Deshalb sollte man darauf hinweisen, daß mehrere Einträge auswählbar sind. Auch ist nicht allen Anwendern klar, wie sie mehrere Einträge selektieren können. Auf modernen PC-Tastaturen geschieht das normalerweise durch Halten der [Strg]-Taste bei gleichzeitigem Anklicken der gewünschten Listeneinträge, beim Macintosh durch gedrückte Umschalttaste.

Einträge vorselektieren

Wenn man nichts anderes angibt, ist zunächst kein Eintrag einer Auswahlliste vorselektiert. Es läßt sich ein Eintrag vorselektieren. In Verbindung mit Mehrfachauswahl können auch mehrere Einträge vorselektiert werden. Vorselektierte Einträge haben einen sichtbaren Markierungsbalken.

```
Wen kennen Sie am besten?
<P><SELECT NAME="BestKenn" SIZE=3>
<OPTION> Rosa Luxemburg
<OPTION SELECTED> Helmut Kohl
<OPTION> Heino
</SELECT>
```

Auswahllisten mit Vorselektion werden wie gewöhnliche Auswahllisten definiert. Um einen Eintrag der Auswahlliste vorzuselektieren, gibt man im Tag <option> des betreffenden Eintrags den Zusatz »selected« an.

Absendewert von Einträgen bestimmen

Normalerweise wird beim Absenden des Formulars der Text eines ausgewählten Listeneintrags übertragen, der hinter <option> steht. Man kann aber bestimmen, daß intern ein anderer Text übertragen wird.

```
Ihre Pizza-Bestellung:
<P>
<SELECT NAME="Pizza" SIZE=5>
<OPTION VALUE="P101"> Pizza Napoli
<OPTION VALUE="P102"> Pizza Funghi
<OPTION VALUE="P103"> Pizza Capriciosa
<OPTION VALUE="P104"> Pizza Vegetabile
<OPTION VALUE="P105"> Pizza Mexicana
<OPTION VALUE="P106"> Pizza Quatro Stagioni
<OPTION VALUE="P107"> Pizza de la Casa
<OPTION VALUE="P108"> Pizza Calzone
<OPTION VALUE="P109"> Pizza con tutti
</SELECT>
```

Auswahllisten mit anderen Absendewerten werden wie gewöhnliche Auswahllisten definiert. Um für einen Eintrag dieser Auswahllisten einen anderen Absendewert zu bestimmen, gibt man im Tag <option> des betreffenden Eintrags das Attribut value= an. Die Angabe muß in Anführungszeichen stehen. Beim Absenden des Formulars wird dann der hier bestimmte Text eines ausgewählten Eintrags übertragen, nicht der Text, der dem Anwender beim Listeneintrag angeboten wurde.

Buttons und Boxen

Radiobuttons

Dies ist eine Gruppe von beschrifteten Buttons, von der der Anwender einen markieren kann, und zwar immer nur einen. Der Wert dieses Buttons wird beim Absenden des Formulars übertragen (Dtei »Buttons.html«).

```
Geben Sie Ihre Zahlungsweise an:
<P>
<INPUT TYPE=radio NAME="Zahlmethode" VALUE="Mastercard"> Mastercard
<BR>
<INPUT TYPE=radio NAME="Zahlmethode" VALUE="Visa"> Visa
<BR>
<INPUT TYPE=radio NAME="Zahlmethode" VALUE="AmericanExpress"> American
Express
```

Radiobuttons werden durch <input> definiert. Dahinter folgt die Angabe »type=radio«. Jeder Radiobutton muß einen internen Bezeichnernamen erhalten, und zwar mit dem Attribut name=. Alle Radiobuttons, die den gleichen Namen haben, gehören zu einer Gruppe, d.h. von diesen Buttons kann der Anwender genau einen markieren. Der Name sollte nicht zu lang sein

und darf keine Leerzeichen und keine deutschen Umlaute enthalten. Man verwende als Sonder-zeichen höchstens den Unterstrich »_« und setze den Namen in Anführungszeichen.

Mit dem Attribut value= bestimmt man einen internen Bezeichnerwert für jeden Radiobutton. Wenn der Anwender das Formular abschickt, wird der Bezeichnerwert des markierten Buttons übertragen. Man setzt den Bezeichnerwert in Anführungszeichen.

Hinter dem <input>-Tag wird der Text eingegeben, mit dem der Radiobutton bei der Anzeige im Browser beschriftet sein soll.

Checkboxen definieren

Checkboxen sind eine Gruppe von beschrifteten Buttons, aus der der Anwender keinen, einen oder mehrere »ankreuzen« kann. Die Werte von markierten Checkboxen werden beim Absenden des Formulars mit übertragen.

```
Geben Sie bei der Pizzabestellung die gew&uuml;nschten Zutaten an:
<P>
<INPUT TYPE=checkbox NAME="zutat" VALUE="salami"> Salami
<BR>
<INPUT TYPE=checkbox NAME="zutat" VALUE="pilze"> Pilze
<BR>
<INPUT TYPE=checkbox NAME="zutat" VALUE="sardellen"> Sardellen
```

Checkboxen werden durch <input> definiert. Dahinter folgt die Angabe type="checkbox". Jede Checkbox muß einen internen Bezeichnernamen erhalten, und zwar mit der Angabe name=. Alle Checkboxen, die den gleichen Namen haben, gehören zu einer Gruppe, d.h. von diesen Elementen kann der Anwender keines, eines oder mehrere ankreuzen.
Der Name sollte nicht zu lang sein und darf keine Leerzeichen und keine deutschen Umlaute enthalten, als Sonderzeichen höchstens den Unterstrich »_«. Man setzt den Namen in Anführungszeichen.

Mit dem Attribut value= bestimmt man einen internen Bezeichnerwert für jede Checkbox. Wenn der Anwender das Formular abschickt, werden die Bezeichnerwerte des oder der angekreuzten Buttons übertragen. Den Bezeichnerwert setzt man in Anführungszeichen.

Hinter dem Tag wird der Text eingegeben, mit dem die Checkbox bei der Anzeige im Browser beschriftet sein soll.

Einträge vorselektieren

Wenn Sie nichts anderes angeben, ist bei Radiobuttons und Checkboxen zunächst kein Eintrag vorselektiert. Man kann bei Radiobuttons einen Eintrag und bei Checkboxen einen oder mehrere Einträge vorselektieren.

```
Sind Sie:
<P>
<INPUT TYPE=radio NAME="Geschlecht" VALUE="Mann"> ein Mann?
<INPUT TYPE=radio NAME="Geschlecht" CHECKED VALUE="Frau"> eine Frau?
<P>
Ich mag:
```

```
<INPUT TYPE=checkbox NAME="Vorliebe" CHECKED VALUE="Urlaub"> Urlaub
<INPUT TYPE=checkbox NAME="Vorliebe" CHECKED VALUE="Geld"> Geld
```

Um einen Radiobutton oder eine Checkbox vorzuselektieren, setzt man bei der Definition des Buttons das Attribut »checked« hinzu.

Klick-Buttons

Klick-Buttons definieren (herkömmlich)

Man kann anklickbare Buttons definieren, die keine spezielle Bedeutung haben. Sinnvoll sind solche frei definierbaren Buttons nur in Verbindung mit Scriptsprachen wie JavaScript. In der Regel werden sie dazu verwendet, Verweise oder andere JavaScript-gesteuerte Befehle auszuführen.
Die hier beschriebene Möglichkeit hat den Vorteil, daß sie auch von älteren Browsern (Netscape seit Version 2.x, Explorer seit Version 3.x) interpretiert wird.

```
Wenn bei Ihnen JavaScript funktioniert, hat der folgende Button die gleiche
Bedeutung wie der Back-Button im WWW-Browser:
<P>
<INPUT TYPE=button VALUE="Zur&uuml;ck" onClick="history.back()">
```

Mit <input type=button> definiert man einen Button. Die Beschriftung des Buttons bestimmt man mit der Zusatzangabe value= . Die Angabe muß in Anführungszeichen stehen. Um anzugeben, was passieren soll, wenn der Button angeklickt wird, kann man beispielsweise den JavaScript Event-Handler onClick= verwenden. Hinter dem Istgleichzeichen gibt man einen JavaScript-Befehl ein, z.B. den Aufruf einer selbstgeschriebenen JavaScript-Funktion, oder – wie im Beispiel – einen einfachen JavaScript-Befehl.

Klick-Buttons definieren (HTML 4.0)

Ab HTML 4.0 dürfen anklickbare Buttons endlich so heißen wie sie heißen: nämlich Button. Solche Buttons sind flexibler als herkömmliche Buttons, denn sie dürfen auch einen definierten Inhalt haben, etwa eine Grafik.

Der MS Internet Explorer interpretiert diesen neuen HTML-Befehl ab Version 4.x, Netscape kennt den Befehl in Version 4.x noch nicht.

```
Wenn bei Ihnen JavaScript funktioniert, hat der folgende Button die gleiche
Bedeutung wie der Back-Button im WWW-Browser:
<P>
<BUTTON NAME="Klickmich" TYPE="button" VALUE="go back" onClick=
"history.back()">
<IMG SRC="klick.gif" ALT="Klickbild">
<P>!GO BACK!
</BUTTON>
```

Die Definition eines solchen Buttons leitet man mit <button> ein. Dieses Tag hat ein Abschluß-Tag </button>, mit dem man die Definition des Buttons am Ende abschließt. Zwischen dem einleitenden Tag und dem End-Tag können Inhalte stehen. Alles, was man innerhalb von <button>...</button> notiert, wird als »Beschriftung« des Buttons angezeigt. Das dürfen durchaus auch Grafikreferenzen sein, so wie im obigen Beispiel. Diese Inhalte werden innerhalb der Button-Fläche zentriert ausgerichtet.

Im einleitenden <button>-Tag notiert man verschiedene Angaben zum Button. Etwas komisch erscheint die Angabe »type=button«, wo doch das Tag schon so heißt. Man notiere diese Angabe jedoch bei allen Buttons, die man für Scriptsprachen verwendet. Denn mit Hilfe des <button>-Tags lassen sich auch zwei andere Button-Typen definieren, nämlich Buttons zum Absenden und Abbrechen.

Mit dem Attribut »name« kann man einen Namen für den Button vergeben. Unter diesem Namen ist der Buttons beispielsweise in JavaScript ansprechbar. Mit dem Attribut value= läßt sich eine Beschriftung für den Button bestimmen, falls man keinen Inhalt innerhalb von <button>...</button> notiert. Zu beachten ist jedoch, daß der Explorer 4.0 diese Angabe ignoriert und bei leerem Inhalt einen leeren Button anzeigt.

Um anzugeben, was passieren soll, wenn der Button angeklickt wird, kann man beispielsweise den JavaScript Event-Handler onClick= verwenden. Hinter dem Istgleichzeiten gibt man einen JavaScript-Befehl ein, z.B. den Aufruf einer selbstgeschriebenen JavaScript-Funktion, oder – wie im Beispiel – einen einfachen JavaScript-Befehl.

Grafiken, die als Button-Inhalt angezeigt werden, dürfen kein Attribut usemap= für verweissensitive Flächen enthalten.

Datei-Buttons

Datei-Buttons erlauben dem Anwender, eine Datei von seinem lokalen Rechner zusammen mit dem Formular zu übertragen. Wenn ein CGI-Script die ankommenden Formulardaten auf dem Server-Rechner verarbeitet, ist dadurch dem Anwender das Uploaden (Hochladen) von Dateien auf den Server-Rechner ermöglicht.

```
<FORM ACTION="/cgi-bin/upload.pl" METHOD=post ENCTYPE="multipart/form-
data">
<P>Senden Sie eine Text- oder HTML-Datei!
<INPUT TYPE=file SIZE=50 MAXLENGTH=100000 NAME="Datei"
ACCEPT="text/*">
<BR>
<INPUT TYPE=submit VALUE="Absenden">
</FORM>
```

Mit <input type=file> definieren Sie einen Datei-Button. Der Browser sollte dann ein Eingabefeld anzeigen, das die Eingabe einer Datei (in den meisten Fällen mit Pfadnamen) erlaubt. Rechts daneben sollte der Browser einen Button anzeigen, bei dessen Betätigen ein lokaler Dateiauswahl-Dialog am Bildschirm erscheint. Die Größe des Eingabefeldes (Anzahl Zeichen) kann man mit size= bestimmen.

Gibt man das Attribut maxlength= an, sollte der WWW-Browser die dahinter notierte Zahl als maximal erlaubte Dateigröße in Bytes interpretieren. Im Beispiel wird auf diese Weise die die Bytezahl auf 100.000 Byte begrenzt. Wenn man maxlength= wegläßt, kann der Anwender beliebig große Dateien senden.
Will man nur bestimmte Dateitypen zulassen, kann man mit der Angabe accept= die erlaubten Dateitypen eingrenzen. Hinter dem Istgleichzeichen läßt sich ein Mime-Typ angeben. Dabei ist auch das Wildcardzeichen »*« bei Subtypen erlaubt. Im Beispiel werden alle Textdateien akzeptiert. Dazu gehören reine Textdateien (*.txt), aber auch HTML-Dateien (*.html,*.htm).

Wichtig ist, daß im einleitenden <form>-Tag die Angabe enctype="multipart/form-data" notiert wird, wenn das Formular einen Dateibutton enthält.

Elemente gruppieren

Größere Formulare bestehen häufig aus Gruppen von Elementen. Ein typisches Bestellformular besteht beispielsweise aus Elementgruppen wie »Absender«, »bestellte Produkte« und »Formular absenden/abbrechen«. Solche Elementgruppen kann man ab HTML 4.0 eigens auszeichnen. Ein Browser sollte Elementgruppen durch Linien oder ähnliche Effekte optisch sichtbar machen.

Der MS Internet Explorer interpretiert diese Befehle ab Version 4.x. Netscape kennt die Befehle in der Version 4.x noch nicht vollständig.

```
<FORM>
<FIELDSET>
<LEGEND><B>Absender</B></LEGEND>
<TABLE>
<TR>
<TD ALIGN=right>Ihr Vorname:
<TD><INPUT TYPE=text SIZE=40 MAXLENGTH=40>
</TR>
<TR>
<TD ALIGN=right>Ihr Zuname:
<TD><INPUT TYPE=text SIZE=40 MAXLENGTH=40">
</TR>
</TABLE>
</FIELDSET>

<FIELDSET>
<LEGEND><B>Wunsch</B></LEGEND>
<TABLE>
<TR>
<TD ALIGN=right>Ihr Wunsch:
<TD><INPUT TYPE=text SIZE=40 MAXLENGTH=40>
</TR>
<TR>
<TD ALIGN=right>Ihr Zusatzwunsch:
<TD><INPUT TYPE=text SIZE=40 MAXLENGTH=40">
</TR>
</TABLE>
</FIELDSET>

<FIELDSET>
<LEGEND><B>Formular</B></LEGEND>
<INPUT TYPE=submit VALUE="Absenden">
<INPUT TYPE=reset VALUE="Abbrechen">
</FIELDSET>
</FORM>
```

Eine zusammengehörige Gruppe von Formularelementen schließt man in die Tags <fieldset></fieldset> ein. Innerhalb dieser Tags lassen sich beliebige Teile eines Formulars definieren.

Unterhalb des einleitenden <fieldset>-Tags und vor den ersten Formularinhalten der Gruppe sollte man ferner eine Gruppenüberschrift (Legende) für die Elementgruppe vergeben.

Schließen Sie den Gruppenüberschriftentext in die Tags <legend>...</legend> ein. Den Text innerhalb kann man mit Hilfe von HTML/CSS nach Wunsch formatieren.

Grafische Buttons

HTML stellt zwei Standard-Buttons bei der Formulareingabe zur Verfügung: Im <input«-Tag notiert man das »Type«-Attribut »submit« (absenden, bestätigen) oder »reset« (abbrechen, wörtlich: zurücksetzen). Mit dem attribut »value« wird in Anführungszeichen die Beschriftung des Buttons notiert.

Es lassen sich jedoch auch Grafiken referenzieren und als Absendebuttons verwenden:

```
<FORM ACTION="/cgi-bin/upload.pl" METHOD=post ENCTYPE="multipart/form-
data">
<P>Senden Sie eine Text- oder HTML-Datei!
<INPUT TYPE=file SIZE=50 MAXLENGTH=100000 NAME="Datei"
ACCEPT="text/*">
<BR>
<INPUT TYPE=image SRC="Klick.gif">>
</FORM>
```

Layer (Schichten)

Layer sind ein Element von HTML, das Netscape mit der Version 4.0 seines Browsers einführte. Mit Hilfe der beiden HTML-Tags, die dazugehören, ist es möglich, beliebige Bereiche einer HTML-Datei als exakt positionierte Bereiche auszuzeichnen. So kann man z.B. eine Überschrift, eine Grafik und erklärenden Text dazu als Layer definieren. Diese definierte Einheit läßt man100 Pixel von links und 50 Pixel von oben – gemessen am Browser-Fenster – beginnen. Für die gesamte Einheit wird eine maximale Breite definiert, z.B. 200 Pixel.

In HTML allein haben die Layer jedoch nicht allzuviel Bedeutung, sieht man einmal von der Möglichkeit der absoluten Positionierung von Elementen ab. Der eigentliche Zweck der Layer ist es, die HTML-Grundlage für Dynamisches HTML zu bilden. Beim Dynamischen Positionieren, dem Ansatz für Dynamisches HTML nach Netscape, lassen sich Layer-Bereiche mit Hilfe von JavaScript auf vielfältige Weise verändern. So können Layer-Bereiche über den Bildschirm wandern, auf-/zuklappen, anderen Inhalt annehmen usw.

Man kann beliebig viele Layer innerhalb einer HTML-Datei definieren. Layer können sich dabei auch überlappen. Faßt man bspw. zwei Überschriften mit anschließender Grafik und erklärendem Text zu je einem Layer zusammen, können beide Layer die gleichen Pixelpositionen erhalten. Mit Hilfe von JavaScript läßt sich dann dynamisch einer der Layer aus- und der andere einschalten. Es lassen sich auch beide Layer gleichzeitig anzeigen und man kann bestimmen, welcher über dem anderen liegen soll.
Layer können eigene Hintergrundfarben und sogar eigene Hintergrundbilder (Wallpapers) haben. Dadurch ergeben sich viele zusätzliche Gestaltungsmöglichkeiten beim Verteilen von Farben und Formen am Bildschirm.

Wenn man eine GIF-Grafik mit transparentem Hintergrund als eigenen Layer definiert, kann die Grafik über oder unter Texte oder andere Grafiken gelegt und dadurch faszinierende Effekte erzeugt werden.

(Das einzige Problem mit den Layern ist: Sie kamen zu spät. Das W3-Konsortium hatte sich bereits darauf eingeschossen, die Möglichkeit des absoluten Positionierens nicht in HTML, sondern in den CSS Style-Sheets zu verankern. Auch für ein HTML-Tag, das vor allem für das Behandeln mit JavaScript gedacht ist, zeigte man dort wenig Verständnis. In den HTML-Sprachstandard 4.0 fanden die hier vorgestellten Befehle deshalb keinen Einlaß. Angesichts der Tatsache, daß die neuen CSS-Befehle zum Positionieren von Elementen offizieller Standard sind und sowohl von Netscape als auch vom Explorer interpretiert werden, ist die Layer-Technik in HTML ins Abseits geraten. Wegen der starken Verknüpfung ihr und Dynamischem HTML ist dadurch freilich auch der gesamte Netscape-Ansatz für Dynamisches HTML wackelig geworden.)
(Datei »Layer1.html«)

Fest positionierte Layer definieren

Man kann innerhalb des Körpers einer HTML-Datei fest positionierte Layer definieren. Solche Layer haben eine definierte obere linke Ecke im Anzeigefenster. Außerdem läßt sich eine bestimmte Breite erzwingen.

```
<LAYER ID="Beispiel" LEFT=300 TOP=250 WIDTH=150 HEIGHT=400>
<H1>Eine &Uuml;berschrift</H1>
<IMG SRC="Beispiel.gif" ALT="Ein Bild">
</LAYER>
```

Mit <layer> leitet man die Definition eines Layer-Bereichs innerhalb einer HTML-Datei ein. Mit der Angabe name= kann einen Namen für den definierten Bereich vergeben werden.

Mit der Angabe left= bestimmt man die Anzahl Pixel vom linken Rand des Anzeigefensters, mit top= die Anzahl Pixel vom dessen oberen Rand. Mit width= wird eine Anzeigebreite für den Layer-Bereich erzwungen, mit height= die Höhe. Elemente innerhalb des Layers werden dann so umbrochen, daß die angegebene Breite eingehalten wird.

Mit id= wird ein Name für den Layer-Bereich vergeben. Unter diesem Namen kann man den Layer in JavaScript mit Hilfe des Objekts »layers« ansprechen. Neben id= wird bei Layern auch die Angabe name= für diesen Zweck interpretiert.

Zwischen dem einleitenden <layer> und dem erforderlichen, abschließenden </layer> lassen sich beliebige andere Elemente einer HTML-Datei notieren, also zum Beispiel Überschriften, Textabsätze, Tabellen, Grafik- oder Multimedia-Referenzen, Verweise usw.
Alle Elemente, die man innerhalb eines Layers definiert, verhalten sich entsprechend der Eigenschaften, die für den entsprechenden Layer definiert wurde. Den Layer selbst muß man sich dabei vorstellen wie ein eigenes Anzeigefenster ohne sichtbare Umrandungen, das in das normale Anzeigefenster des WWW-Browsers eingebettet ist.
Neben den Angaben zu »left« und »top« gibt es auch noch die ähnlichen Angaben pagex= (wirkt im Normalfall wie left=) und pagey= (wirkt im Normalfall wie top=). Der Unterschied ist, daß »left« und »top« sich genau genommen am Dokumentrand orientieren und nicht am Fensterrand. Der Unterschied macht sich dann bemerkbar, wenn man Layer verschachteln. Dort wird der Unterschied der Attribute näher beschrieben.

Wenn man mit Layern arbeitet und diese absolut im Anzeigebereich positioniert, wird es in den meisten Fällen sinnvoll sein, innerhalb
einer HTML-Datei alle anzuzeigenden Elemente innerhalb von fest positionierten Layern zu notieren. Denn andernfalls gerät einem die Anzeige leicht außer Kontrolle.

Wenn es der Inhalt erfordert, werden Angaben zur Breite außer kraft gesetzt. Wenn beispielsweise eine Grafik oder eine Tabelle mehr Platz in Anspruch nimmt als die angegebene Breite, wird die Grafik bzw. die Tabelle in jedem Fall in voller Breite angezeigt. Dies läßt
sich allerdings verhindern – siehe Anzeigebereich von Layern beschneiden.

Die Angaben zur linken und oberen Ecke sowie zur Breite sind zwar in den meisten Fällen erwünscht, jedoch nicht zwingend erforderlich. Wenn man keine Angaben dazu macht, beginnt der Layer einfach oben links, so wie das erste Element im Dateikörper der HTML-Datei.

Wenn Layer nichts anderes enthalten als GIF-Grafiken mit transparentem Hintergrund, kann man diese Grafiken mit Hilfe der Layer beliebig übereinanderschichten und dadurch interessante Effekte erzeugen.

Inline-Layer definieren

Inline-Layer beginnen relativ an der Stelle, an der sie innerhalb der HTML-Datei stehen. Solche Layer erzeugen keinen eigenen Absatz im Fließtext. Ansonsten können sie aber alle Eigenschaften von Layern haben, also beispielsweise eine eigene Hintergrundfarbe.

```
Das ist<ILAYER>Layer-Text</ILAYER>mitten im Text.
```

Mit <ilayer> wird die Definition eines Inline-Layer-Bereichs eingeleitet. Mit der Angabe name= läßt sich ein Namen für den definierten Bereich vergeben.

So wie im obigen Beispiel definiert, bewirkt ein <ilayer>-Bereich rein gar nichts. Erst durch Angaben zu Hintergrundfarben und Hintergrundbild lassen sich dabei interessante Effekte erzielen.

Man kann auch Attribute wie top= oder left= auf einen <ilayer>-Bereich anwenden. Die Angaben werden dann jedoch nicht als Werte gemessen vom Fensterrand interpretiert, sondern als Werte gemessen von der normalen Position des Inline-Layers. So kann auch z.B. mitten in einem Text eine Anweisung notiert werden wie:

```
Das ist<ILAYER TOP=25 BGCOLOR=„#FF0000“>Layer-Text</ILAYRER>mitten im Text
```

Dieses Wort erscheint um genau 25 Pixel tiefer als der übrige Text und ist rot unterlegt.

Anzeigebereich von Layern beschneiden

Man kann einen Bereich erzwingen, innerhalb dessen der Inhalt eines Layers angezeigt werden muß. Wenn der Inhalt mehr Platz in Anspruch nimmt (z.B. weil eine Grafik breiter ist), wird er abgeschnitten. Mit diesem Merkmal lassen sich Effekte erzeugen, daß der Inhalt eines Layers wie durch ein Fenster gesehen erscheint, wobei beim Blick durch das Fenster nur ein Ausschnitt des Inhalts sichtbar ist. (Datei: »Layer2.html«)

```
<LAYER NAME="Beispiel" LEFT=30 TOP=20 WIDTH=260 CLIP="20,30,200,300">
<H1>Eine &Uuml;berschrift</H1>
<IMG SRC="Beispiel.gif" ALT="Ein Bild">
</LAYER>
```

Mit dem Attribut clip= kann man einen Anzeigebereich erzwingen. Hinter dem Istgleichzeichen notiert man zwei oder vier Zahlenwerte zum Bestimmen des gewünschten Bereichs. Die Angabe muß in Anführungszeichen stehen. Trennen Sie die Zahlenwerte durch Kommata.

Gibt man vier Zahlenwerte an, bedeuten die Zahlen die Koordinaten des gewünschten Rechtecks (in dieser Reihenfolge):
clip=„Pixel für links, Pixel für oben, Pixel für rechts, Pixel für unten“
und zwar relativ zur oberen linken Ecke des Layers. Wenn also im Beispiele etwa durch left=30 und top=20 eine absolute obere linke Ecke für den Layer definiert wird, so bedeutet etwa die erste Angaben bei clip= (20) so viel wie: der erzwungene Anzeigebereich beginnt 20 Pixel links vom linken Rand des Layers (der bei 30 Pixeln liegt), absolut gesehen also bei 50 Pixeln.

Werden zwei Zahlenwerte angeben, bedeuten die Zahlen die Breite und Höhe des gewünschten Rechtecks (in dieser Reihenfolge):
clip=„Pixel für Breite, Pixel für Höhe“ und zwar beginnend bei der oberen linken Ecke des Layers.

Layer mit Inhalt aus einer anderen Datei

Es ist möglich, innerhalb eines Layers eine andere HTML-Datei anzuzeigen:

```
<LAYER SRC="andere.html" TOP=150 LEFT=200</LAYER>
```

Mit dem Attribut »src« läßt sich eine andere HTML-Datei in einen Layer einbinden. Die Datei nimmt dabei den üblichen Raum im Anzeigefenster ein, beginnend bei den angegebenen Positionen zu »top« und »left«. Angaben zu »width« und »height« werden nur eingehalten, wenn der Inhalt der Datei in Breite und Höhe weniger Platz in Anspruch nimmt.

Layer verschachteln

Sie können innerhalb von Layer-Bereichen andere Layer definieren.

Beispiel 1:

```
<LAYER TOP=100 LEFT=100>
<LAYER TOP=100 LEFT=100>
<IMG SRC="bild.gif">
</LAYER>
</LAYER>
```

Beispiel 2:

```
<LAYER TOP=100 LEFT=100>
<LAYER PAGEY=100 PAGEX=100>
<IMG SRC="bild.gif">
</LAYER>
</LAYER>
```

Innerhalb eines Layers kann man andere Layer definieren.

Wichtig ist dabei die unterschiedliche Wirkung der Positionsangaben. Im Beispiel 1 erhält der innere Layer die gleichen Angaben zu »left« und »top« wie der äußere Layer. Der äußere Layer ist dabei der Bezugspunkt für die Startposition des inneren Layers, nicht das Anzeigefenster! Der innere Layer beginnt im Beispiel 1 also 100 Pixel von oben und 100 Pixel von links, gemessen an der linken oberen Ecke des äußeren Layers.

Ganz anders dagegen im Beispiel 2. Dort erhält der innere Layer die Angaben pagey=100 und pagex=100 (pagex = Seite x-Wert, pagey = Seite y-Wert). Diese Angaben bewirken, daß als Bezugspunkt das Anzeigefenster und nicht der übergeordnete Layer verwendet wird. Im Beispiel 2 hat das zur Folge, daß der äußere und der innere Layer genau an der gleichen Position beginnen.
Da der äußere Layer in den Beispielen keinen anderen Inhalt als den inneren Layer hat, ist am Bildschirm nur der innere Layer zu sehen.

Hintergrund

Wie bei anderen HTML-Dateien und –Elementen, kann auch der Hintergrund eines Layers bzw. eines iLayers zugewiesen werden.
Für eine Farbe wird im »Layer«Tag das entspr. Attribut »BGCOLOR=„#xxxxxx“« (hier ist ggf. an die »FONT COLOR=« zu denken!) zugewiesen, für ein Hintergrundbild das Attribut »BACKGROUND=„Datei.gif“«.

Angaben zur Schichtposition

Wenn man mehrere Layer in einer HTML-Datei definiert, deren Anzeigebereiche sich überlappen, werden die Layer normalerweise in der Reihenfolge übereinander angezeigt, in der sie definiert werden. Man kann jedoch die Reihenfolge ändern und angeben, welcher andere Layer unmittelbar über oder unter dem aktuellen Layer liegen soll. Auf diese Weise läßt sich das Überdecken von Layern kontrollieren. Voraussetzung ist, daß für jeden Layer ein Name vergeben wird:

```
<LAYER ID=“Erster" LEFT=100 TOP=100>
 <IMG SRC="datei1.gif" ALT="Ein Bild">
 </LAYER>
 <LAYER ID="Zweiter" LEFT=150 TOP=150 ABOVE="Erster">
 <IMG SRC="datei2.gif" ALT="Ein anderes Bild">
 </LAYER>
 <LAYER ID="Dritter" LEFT=200 TOP=200 BELOW="Erster">
 <IMG SRC="datei3.gif" ALT="Ein ganz anderes Bild">
 </LAYER>
```

Mit dem Attribut »above« im einleitenden <layer>- oder <ilayer>-Tag kann man bestimmen, daß ein zuvor definierter, benannter Layer unmittelbar über dem aktuellen Layer liegt. Das heißt, der aktuelle Layer soll bei Überschneidung von dem genannten Layer überdeckt werden.
Mit dem Attribut »below« im einleitenden <layer>- oder <ilayer>-Tag läßt sich bestimmen, daß ein zuvor definierter, benannter Layer unmittelbar unter dem aktuellen Layer liegt. Das heißt, der aktuelle Layer soll bei Überschneidung den genannten Layer überdecken.
Die Layernamen hinter »above« und »below« sollten in Anführungszeichen stehen.

Bei »above« und »below« dürfen nur Namen von Layern angeben werden, die in der Datei bereits zuvor definiert wurden. Die Angabe von Layern, die erst nach dem aktuellen Layer definiert werden, kann zu Anzeigefehlern führen. Im ersten definierten Layer einer HTML-Datei sollten deshalb keine Angaben zu »above« oder »below« vorkommen.

Es ist nur je eine Angabe für »above« und »below« bei der Definition eines Layers erlaubt. Denn mit diesen Angaben legt man immer nur fest, welcher andere Layer direkt unter oder über dem aktuellen Layer angezeigt werden soll.

Man kann die Reihenfolge auch ändern, indem man für die einzelnen Layer Nummern vergibt. Layer mit höherer Nummer überdecken Layer mit niedrigerer Nummer:

```
<LAYER NAME="Erster" LEFT=100 TOP=100 Z-INDEX=2>
<IMG SRC="datei1.gif" ALT="Ein Bild">
</LAYER>
<LAYER NAME="Zweiter" LEFT=150 TOP=150 Z-INDEX=1>
<IMG SRC="datei2.gif" ALT="Ein anderes Bild">
</LAYER>
<LAYER NAME="Dritter" LEFT=200 TOP=200 Z-INDEX=3>
<IMG SRC="datei3.gif" ALT="Ein ganz anderes Bild">
</LAYER>
```

Mit dem Attribut »z-index« im einleitenden <layer>- oder <ilayer>-Tag bestimmt man die Reihenfolge der Layer bei Überlappungen. Man gibt bei jedem definierten Layer z-index= und dahinter eine Nummer an. Der Layer mit der höchsten Nummer überdeckt alle anderen. Der Layer mit der niedrigsten Nummer wird von allen anderen überdeckt.

Bei den Angaben zu »z-index« sind auch negative Werte erlaubt. Dadurch bewirkt man bei verschachtelten Layern, daß Layer, die innerhalb eines anderen Layers definiert werden, von ihren »Eltern«-Layern überdeckt werden. Bei positiven Zahlen ist dies umgekehrt.

Layer verstecken und anzeigen

Es läßt sich bei der Definition von Layern festlegen, ob diese zunächst angezeigt werden sollen oder nicht. Das ist von Bedeutung, wenn man die Anzeige der Layer mit Hilfe von JavaScript dynamisch ein-/ausschalten will.

```
<LAYER NAME="Erster" LEFT=100 TOP=100 VISIBILITY=show>
<IMG SRC="datei1.gif" ALT="Ein Bild">
</LAYER>
<LAYER NAME="Zweiter" LEFT=120 TOP=200 VISIBILITY=hide>
<IMG SRC="datei2.gif" ALT="Ein anderes Bild">
</LAYER>
```

Mit dem Attribut »visibility« im einleitenden <layer>- oder <ilayer>-Tag kann man die Anzeige des Layers explizit erzwingen oder verhindern.
Mit »visibility=show« legt man fest, daß der Layer in jedem Fall angezeigt wird.
Mit »visibility=hide« wird festgelegt, daß der Layer nicht angezeigt, also versteckt wird.
Mit »visibility=inherit« legt man fest, daß der Layer nur dann angezeigt wird, wenn sein »Eltern«-Element ebenfalls angezeigt wird. Diese Angabe ist nur bei inneren Layern in verschachtelten Layern von Bedeutung.

Style-Sheet-Bereiche

Zentrale Style-Sheet-Bereiche definieren

Um zentrale Style-Sheets (Formatiermöglichkeiten für HTML-Tags) zu definieren, kann man im Kopf einer HTML-Datei einen oder mehrere Style-Sheet-Bereiche definieren.(Datei »Sheet-Bereiche.html«)

```
<HEAD>
 <TITLE>Sheet-Bereich.html</TITLE>
 <STYLE TYPE="text/css">
 <!--
  body { margin:2cm }
  h1 { font-size:24pt }
 //-->
 </STYLE>
 <STYLE TYPE="text/javascript">
 <!--
 with(tags.H2)
  {
   color = "red";
   fontSize = "16pt";
   marginTop = "2cm";
  }
 //-->
 </STYLE>
 </HEAD>
```

Mit <style> leitet man einen Bereich zum Definieren von Formaten ein. Innerhalb des einleitenden <style>-Tags kann man mit dem Attribut type= angeben, welche Style-Sheet-Sprache man innerhalb des Bereichs zum Definieren der Formate benutzen will. Die gängigste Angabe ist dabei type="text/css". Mit dieser Angabe definiert man CSS Style-Sheets als Sprache für die Formatdefinitionen. Die Angabe sollte in Anführungszeichen stehen.

Man kann aber auch andere Sprachen benutzen. Falls man mehrere Style-Sprachen innerhalb der gleichen HTML-Datei einsetzen möchte, definiert man einfach für jede Sprache einen <style>-Bereich. Im Beispiel etwa werden zwei solcher Bereiche definiert.
Der erste Bereich enthält CSS-Formatdefinitionen, der zweite Formatdefinitionen in der Netscape-spezifischen Sprache JSSS Style-Sheets.

Mit </style> wird ein Abschnitt für Formatdefinitionen beendet.

Es ist empfehlenswert, den Inhalt von Style-Sheet-Bereichen in einen mehrzeiligen Kommentar (<!-- bzw. //-->) zu setzen, so wie im Beispiel. Dadurch verhindert man, daß ältere Browser die Inhalte irrtümlich als Text anzeigen.

Lokale Style-Sheet-Bereiche definieren

Für Schnellformatierung im Text mit Hilfe von CSS Style-Sheets gibt es ein eigenes HTML-Tag: (Datei: »Span.html«)

```
Dies ist ein <span style="font-size:18pt">formatierter Text</span>
```

Mit kann man einen Bereich für eine Formatierung definieren. Das Tag für sich genommen hat keine eigene Bedeutung und bewirkt keinen optisch sichtbaren Effekt. Zum Leben erwacht es erst, wenn man im einleitenden -Tag das Universalattribut style= notiert. Dahinter folgen, in Anführungszeichen stehend, eine oder beliebig viele Style-Sheet-Angaben, die durch ein Semikolon voneinander abgetrennt werden. Zu bachten ist, daß bei den Attributen ein Doppelpunkt statt eines Istgleich-Zeichens zu setzen ist!

Kommentare

HTML beinhaltet die Möglichkeit, an beliebigen Stellen innerhalb einer Datei Kommentare einzufügen. Kommentare werden von Browsern ignoriert, d.h. bei der Präsentation nicht angezeigt. Kommentare sind z.B. sinnvoll, um interne Angaben zu Autor und Erstelldatum in einer Datei zu plazieren, um interne Anmerkungen zu bestimmten Textstellen zu machen, oder um verwendete HTML-Befehle intern auszukommentieren.

Beispiel 1:

```
<!-- Dieser Text ist ein Kommentar -->
```

Beispiel 2:

```
<!-- Erste Zeile eines mehrzeiligen Kommentars
Letzte Zeile des Kommentars //-->
```

Kommentare werden durch die Zeichenfolge <!-- eingeleitet. Dahinter folgt beliebig langer Kommentartext. Darin kann man auch HTML-Befehle erwähnen, ohne diese maskieren zu müssen. Ein einzeiliger Kommentarbereich durch die Zeichenfolge --> beendet, ein mehrzeiliger Kommentarbereich durch //-->.

Bei einigen neueren HTML-Befehlen, vor allem beim Einbinden von JavaScript, wird empfohlen, den Inhalt in Kommentare zu setzen, da einige ältere Browser darüber nicht hinweglesen, sondern den darin enthaltenen Text/Programmcode am Bildschirm ausgeben statt ihn zu überlesen.

Während man an seinen Dateien bastelt, wird häufig verschiedene HTML-Befehle, Farbangaben usw. testen und ausprobieren. Dabei ist es hilfreich, bisherige Lösungen vor dem Ausprobieren einer neuen Lösung nicht einfach zu überschreiben, sondern in einen Kommentar-Befehl zu setzen. So muß man sich nicht ärgern, wenn man etwas überschreibt, das eigentlich doch besser war als das, womit man es überschrieben hat.

Script-Bereich definieren

Man kann im Dateikopf oder auch innerhalb des Dateikörpers einer HTML-Datei Script-Bereiche definieren. Innerhalb von Script-Bereichen lassen sich Befehle einer Scriptsprache wie JavaScript notieren. (Datei:»Script-Bereiche.html«)

```
<HEAD>
<TITLE>Text des Titels</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
UserName = window.prompt("Dein Vorname:","Vorname");
//-->
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!--
document.write("<h1>Hallo " + UserName + "!</h1>");
//-->
</SCRIPT>
</BODY>
</HTML>
```

Mit <script> leitet man einen Script-Bereich ein. Innerhalb des einleitenden <script>-Tags gibt man mit dem Attribut »anguage« an, welche Script-Sprache man innerhalb des Bereichs benutzen möchte. Die gängigste Angabe ist dabei language="JavaScript". Andere denkbare Sprachangaben sind beispielsweise JScript oder VBScript (beide Microsoft).
Auch die Versionsnummer der Sprache ist erlaubt, beispielsweise JavaScript1.2. Die Angabe sollte in Anführungszeichen stehen.
Mit </script> beendet man einen Abschnitt für Formatdefinitionen.

Man kann durchaus mehrere Script-Bereiche definieren. Im sind es zwei: einer im Dateikopf und einer im Dateikörper.

Im obigen Beispiel wird beim Einlesen der Datei ein Dialogfenster geöffnet, in dem der Anwender seinen Vornamen eingeben kann. Dies passiert im ersten der beiden Script-Bereiche. Der eingegebene Vorname wird dann dynamisch in die Datei geschrieben. Der entsprechende Befehl dazu steht im zweiten Script-Bereich.

Es ist empfehlenswert, den Inhalt von Script-Bereichen in einen mehrzeiligen Kommentar (<!-- bzw. //-->) zu setzen, so wie hier. Dadurch verhindert man, daß ältere Browser die Inhalte irrtümlich als Text anzeigen.

Noscript-Bereich definieren

Man kann auch einen Bereich definieren, der nur angezeigt wird, wenn die verwendete Script-Sprache nicht verfügbar ist, das Script also nicht ausführbar ist. Dies ist der Fall, wenn der Browser die Scriptsprache nicht kennt, oder wenn der Anwender das Interpretieren der Scriptsprache in den Einstellungen seines Browsers ausgeschaltet hat.

Wichtig ist eine solche Angabe beispielsweise, wenn Ihre Seiten intensiv JavaScript benutzen, um Inhalte anzuzeigen oder Verweise auszuführen. In solchen Fällen ist ein Projekt ohne JavaScript kaum nutzbar. Mit einem Noscript-Bereich kann man einen entsprechenden Hinweis einbauen.

```
<HEAD>
<TITLE>Text des Titels</TITLE>
<script language="JavaScript">
<!--
```

```
UserName = window.prompt("Dein Vorname:","Vorname");
//-->
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!--
document.write("<h1>Hallo " + UserName + "!</h1>");
//-->
</SCRIPT>
<NOSCRIPT>
<B>Dieser Text wird angezeigt, wenn JavaScript nicht funktioniert</B>
</NOSCRIPT>
</BODY>
</HTML>
```

Mit <noscript> leitet man einen Noscript-Bereich ein, mit </noscript> wird erbeendet. Dazwischen kann man einfach Text notieren, aber auch HTML-Formatierungen und andere HTML-Elemente verwenden.
Browser, die das <noscript>-Tag kennen, zeigen den Inhalt dazwischen nur dann an, wenn der Anwender die benutzte Scriptsprache, etwa JavaScript, deaktiviert hat. Browser, die gar keine Scriptsprachen kennen, kennen zwar normalerweise auch das <noscript>-Tag nicht, aber einer alten Regel gemäß ignorieren sie das Tag einfach und zeigen den Inhalt des »Noscript«-Bereichs ganz normal an.

Weitere Angaben zum Script

Es gibt in HTML eine Reihe weiterer Optionen für Script-Bereiche.

```
<SCRIPT LANGUAGE="JavaScript" SRC="tuwas.js" TYPE="text/javascript"
DEFER></SCRIPT>
```

Mit »src« kann man ein Script einbinden, das in einer separaten Datei notiert ist. Das ist von Vorteil, wenn man ein Script auf vielen Seiten verwenden will. Dann braucht man das Script nur noch zu referenzieren, statt es jedesmal in der Datei zu notieren. Der Script-Bereich in der Datei kann in diesem Fall leer bleiben, so wie im Beispiel. Er kann aber auch noch ein lokales Script enthalten.

Wenn man mit »src« eine separate Scriptdatei einbindet, empfiehlt sich außerdem die Angabe des zugehörigen Mime-Types. Dazu dient das Attribut »type«. Wenn man mit JavaScript arbeitet, gibt man hierbei type="text/javascript" an.

Das Attribut »defer« soll verhindern, daß ein Script dynamisch Text schreibt. Einzelne JavaScript-Befehle wie »document.write()« oder auch typische Befehle bei Dynamischem HTML werden dadurch unterbunden.

Dynamische grafische Buttons (Rollover-Effekt)

Auf WWW-Seiten kommen häufig grafische Buttonleisten zum Einsatz, die Verweise zu bestimmten Seiten des Projekts enthalten. Wenn der Anwender mit der Maus über solche Grafiken fährt, erkennt er am veränderten Mauscursor und an den Verweiszielanzeigen in der Statuszeile des WWW-Browsers, daß es sich um Verweise handelt. Mit Hilfe von JavaScript

kann man solche grafischen Verweise jedoch noch deutlicher kenntlich machen und der Benutzerführung gleichzeitig ein wenig mehr Pep verleihen.
Dazu braucht man je zwei gleichartige, farblich unterschiedliche Grafiken für je einen Grafikbutton.

Mit Hilfe von JavaScript läßt sich nun eine Grafik durch eine andere ersetzen, zum Beispiel, wenn die Grafik als Verweises dient und der Anwender mit der Maus über die Grafik fährt. Wie das im einzelnen funktioniert, wird im folgenden beschrieben.

Dieses Beispiel funktioniert mit Netscape ab Version 3.x und mit dem MS Internet Explorer ab Version 4.x. Es ist zu beachten, daß es bei Verwendung dynamischer Grafiken in Tabellen innerhalb von Tabellen bei Netscape zu Verschiebungsproblemen kommen kann (Datei »Rollover.html«).

```
<HTML>
  <HEAD>
  <TITLE>Rollover.html</TITLE>
  <SCRIPT LANGUAGE="JavaScript">
   <!--
   1* Normal1 = new Image();
   Normal1.src = "button1a.gif"; /* Hier erste Standard-Grafik angeben */
   Highlight1 = new Image();
   Highlight1.src = "button1b.gif"; /* Hier erste Highlight-Grafik angeben */

   Normal2 = new Image();
   Normal2.src = "button2a.gif"; /* Hier zweite Standard-Grafik angeben */
   Highlight2 = new Image();
   Highlight2.src = "button2b.gif"; /* Hier zweite Highlight-Grafik angeben */

   /* usw. fuer alle weiteren zu benutzenden Grafiken */

   2* function Bildwechsel(Bildnr,Bildobjekt)
   {
    window.document.images[Bildnr].src = Bildobjekt.src;
   }
  //-->
   //-->
  </SCRIPT>
  </HEAD>
  <BODY>

  3* <table><tr><td>
  <A HREF="index.htm" onMouseOver="Bildwechsel(0,Highlight1)"
  onMouseOut="Bildwechsel(0,Normal1)">
  <IMG SRC="button1a.gif"></A>
  </TD><TD>
  <A HREF="verweise.htm" onMouseOver="Bildwechsel(1,Highlight2)"
  onMouseOut="Bildwechsel(1,Normal2)">
  <IMG SRC="button2a.gif"></A>
  </TD></TR></TABLE>

  </BODY>
  </HTML>
```

Für jede Grafik, die man dynamisch anzeigen möchte, muß man eine Instanz des image-Objekts erzeugen. Das gilt sowohl für die Grafiken, die zunächst in der HTML-Datei referenziert werden, als auch für diejenigen, die beim Überfahren mit der Maus dynamisch angezeigt werden sollen.

Dazu wird im Beispiel mit Anweisungen wie Normal1 = new Image(); eine Objektinstanz erzeugt. Nachdem die Objektinstanz erzeugt ist, sind unter dem gewählten Objektnamen, im ersten Fall Normal1, alle Eigenschaften des Objekts ansprechbar. Zunächst enthält das Objekt noch gar keine Daten. Mit der Anweisung Normal1.src = "button1a.gif"; wird dem neuen Grafikobjekt eine Grafikdatei zugewiesen.

Wiederholen Sie die beiden beschriebenen Anweisungen für jede Grafikdatei, die von dynamischen Änderungen betroffen sein wird – und zwar sowohl für Grafiken, die im »Normalfall« angezeigt werden, als auch für Grafiken, die beim Darüberfahren mit der Maus dynamisch angezeigt werden. Man vergebe dabei jedesmal einen neuen Objektnamen.

Der bisherige Code im Beispiel wird beim Einlesen der HTML-Datei direkt ausgeführt, da er nicht in einer Funktion gebunden ist. Die Ausführung dieses Codes bewirkt aber keine sichtbaren Ausgaben und wird vom Anwender gar nicht bemerkt. Das, was am Bildschirm mit Hilfe von JavaScript passieren soll, nämlich das dynamische Austauschen eines Bildes, geschieht in der definierten Funktion Bildwechsel().

Die Funktion soll aufgerufen werden, wenn der Anwender mit der Maus über einen grafischen Verweis fährt und wenn er den Verweisbereich mit der Maus wieder verläßt. Dazu benötigt die Funktion zwei Parameter: die wievielte Grafik in der Datei ausgetauscht werden soll (Parameter Bildnr), und durch welches zuvor definierte Grafikobjekt das Bild ersetzt werden soll
(Parameter Bildobjekt). Die Funktion kommt dann mit einer einzigen Anweisung aus. Diese Anweisung ersetzt das vorhandene Bild durch das neue. Man beachte hier den wichtigen Zusammenhang: Das dynamische Ersetzen einer Grafik ist nur möglich, wenn für die neue Grafik zuvor eine Instanz des Grafikobjekts erzeugt wurde. Im Beispiel geschah dies ja ganz am Anfang des Scripts.

Damit der gewünschte Effekt zustande kommt, muß man in der HTML-Datei Grafiken als Verweise definieren. Bei Buttons, die nebeneinander liegen sollen, empfiehlt sich eine blinde Tabelle, wie sie auch im Beispiel verwendet wird.

Im einleitenden Verweis-Tag werden die Event-Handler onMouseOver= und onMouseOut= notiert. Bei onMouseOver= (wenn der Anwender mit der Maus über den verweis-sensitiven Bereich, hier eine Grafik fährt) wird die im Script-Bereich definierte Funktion Bildwechsel() aufgerufen, ebenso bei onMouseOut= (wenn der Mauszeiger den verweis-sensitiven Bereich wieder verläßt).

Beim Aufruf von Bildwechsel() werden jeweils die beiden benötigten Parameter übergeben. Man zähle dazu die referenzierten Grafikdateien in der HTML-Datei, aber fange bei 0 an zu zählen, d.h. 0 für die erste Grafik in der Datei, 1 für die zweite Grafik usw. Übergeben Sie den Indexwert der Grafik, die in den Verweis eingebettet ist. Im Beispiel sind in der gesamten HTML-Datei keine anderen als die dynamisch zu tauschenden Grafiken referenziert. Deshalb wird im ersten Fall 0 (erste Grafik der Datei) übergeben, im zweiten Fall 1 (zweite Grafik der Datei).
Beim zweiten Parameter übergeben Sie den Objektnamen für das gewünschte Grafikobjekt. Das ist einer der Namen, die Sie am Anfang des Script-Bereichs vergeben haben. Im Beispiel

wird im ersten Fall z.B. Highlight1 bei onMouseover übergeben, und Normal1 bei onMouseout.

Style-Sheets

Sinn und Zweck

Style-Sheets sind eine unmittelbare Ergänzung zu HTML. Es handelt sich dabei um eine Sprache zur Definition von Formateigenschaften einzelner HTML-Befehle. Mit Hilfe von Style-Sheets kann man z.B. bestimmen, daß Überschriften 1. Ordnung eine Schriftgröße von 18 Punkt haben, in der Schriftart Helvetica, aber nicht fett erscheinen, und mit einem Abstand von 1,75 Zentimeter zum darauffolgenden Absatz versehen werden. Angaben dieser Art sind mit herkömmlichem HTML nicht möglich.

Man kann auch beliebige Bereiche einer HTML-Datei mit einer eigenen Hintergrundfarbe, einem eigenen Hintergrundbild (Wallpaper) oder mit diversen Rahmen versehen. Und ebenfalls beliebige Elemente, sei es eine Grafik, ein Textabsatz, eine Tabelle oder ein Bereich aus mehreren solcher Elemente, pixelgenau im Anzeigefenster des Browsers positionieren. Für Drucklayouts stehen Befehle zur Definition eines Seitenlayouts bereit. Für die akustische Wiedergabe von HTML-Dateien gibt es ein ganzes Arsenal an Befehlen, um künstliche Sprachausgabesysteme feinzusteuern. Spezielle Filter schließlich, die derzeit allerdings noch rein Microsoft-spezifisch sind, erlauben Grafik-Effekte bei normalen Texten, die aus Grafikprogrammen wie PhotoShop bekannt sind.

Ein weiteres wichtiges Leistungsmerkmal von Style-Sheets ist es, daß man Definitionen zentral angeben kann. So beispielsweise im Kopf einer HTML-Datei zentrale Definitionen zum Aussehen einer Tabellenzelle notieren. Alle Tabellenzellen der entsprechenden HTML-Datei erhalten dann die Formateigenschaften, die einmal zentral definiert sind. Das spart Kodierarbeit
und macht die HTML-Dateien kleiner. Style-Sheet-Definitionen lassen sich sogar in separaten Dateien notieren. Solche Style-Sheet-Dateien kann man in beliebig vielen HTML-Dateien referenzieren. Auf diese Weise werden für große Projekte einheitliche Layouts entworfen. Mit ein paar kleinen Änderungen in einer zentralen Style-Sheet-Datei kann man somit für hunderte
von HTML-Dateien ein anderes Layout bewirken.

Style-Sheets unterstützen also erstens die professionelle Gestaltung beim Web-Design, und zweitens helfen sie beim Corporate Design für große Projekte oder für firmenspezifsche Layouts.

Style-Sheet-Sprachen, Versionen und Informationen

Es gibt mehrere Sprachen zum Definieren von Style-Sheets. Die bekannteste ist die CSS-Sprache. CSS steht für »Cascading Style Sheets«. Diese Sprache wird vom W3-Konsortium vorgeschlagen. Die CSS-Sprache ist optimal auf HTML abgestimmt.
Genau so, wie es HTML-Sprachversionen gibt, gibt es auch CSS-Sprachversionen. Die erste Version 1.0 enstand 1996. Seit März 1998 gibt es die Version 2.0.

Style-Sheets und WWW-Browser

Netscape 4.x interpretiert fast den vollen Sprachumfang der CSS-Version 1.0 und einen Teil der Befehle der CSS-Version 2.0. Der Explorer kennt die CSS-Version 1.0 bereits seit seiner Produktversion 3.0. In der Version 4.0 interpretiert er einen Teil der CSS-Version 2.0 und einige spezielle, von Microsoft eingeführte Style-Sheet-Angaben.

Style-Sheets erlauben völlig neue Möglichkeiten beim Webseiten-Layout. Je intensiver man seine Layouts jedoch von Style-Sheet-Effekten abhängig macht, desto langweiliger, enttäuschender und evtl. auch fehlerhaft wirken die Seiten bei Anwendern, die keinen Style-Sheet-fähigen Browser benutzen. Da Netscape 3.x immer noch als wichtige Meßlatte für die Anzeigequalität von WWW-Seiten gilt, in dieser Version aber noch keine Style-Sheets kennt, werden Style-Sheets bislang noch vorsichtig eingesetzt. Allmählich muß man sich jedoch nicht mehr entschuldigen, wenn man Style-Sheets einsetzt. Zumindest gegen den Einsatz der Befehle der CSS-Version 1.0 ist eigentlich nichts mehr. Neuere Befehle sollte man dagegen noch mit Vorsicht anwenden. In jedem Fall die »gestylten« Web-Seiten immer auch mal mit Netscape 3.x testen.

Style-Sheets in einer HTML-Datei definieren

Man kann innerhalb einer HTML-Datei einen Bereich für Style-Sheet-Angaben definieren.

```
<HTML>
 <HEAD>
 <TITLE>Titel der Datei</TITLE>
 <STYLE TYPE="text/css">
 <!--
 /*  ... Style-Sheet-Angaben ... */
 //-->
 </STYLE>
 </HEAD>
 <BODY>
 </BODY>
 </HTML>
```

Mit dem Befehl <style...></style> im Kopf der HTML-Datei definiert man einen Bereich für Style-Sheet-Angaben. Im einleitenden <style>-Tag muß der Typ der Formatdefinition angeben werden. Das geschieht bei allen hier erläuterten Formatiermöglichkeiten durch die Angabe TYPE="text/css". Zwischen dem einleitenden und dem abschließenden Tag kann man dann die eigentlichen Style-Sheet-Angaben definieren.

Damit Browser, die keine Style-Sheets kennen, die Style-Sheet-Angaben nicht irrtümlich als anzuzeigenden Text interpretieren, kann man den Bereich der eigentlichen Style-Sheet-Angaben in einen mehrzeiligen HTML-Kommentar <!-- ... //--> einbinden, so wie im Beispiel.

Die Style-Sheet-Angaben, die auf diese Weise im Kopf einer HTML-Datei definiert werden, sind für diese eine HTML-Datei gültig.

Style-Sheets in separater Datei definieren

In vielen Fällen wird man einheitliche Formate für alle HTML-Dateien eines Projekts haben wollen. In diesem Fall braucht man die Angaben nicht in jeder Datei zu wiederholen. Stattdessen lassen sich die Style-Sheet-Angaben in einer separaten Textdatei notieren und diese Datei einfach in jeder gewünschten HTML-Datei einbinden. Werden die Angaben in der separaten Datei geändert, wirken sich die Änderungen einheitlich auf alle Dateien aus, in denen diese separate Datei eingebunden ist.

```
<HTML>
 <HEAD>
 <TITLE>Titel der Datei</TITLE>
 <LINK REL=stylesheet TYPE="text/css" HREF="formate.css">
 <STYLE TYPE="text/css">
 <!--
  ... Extra-Style-Sheet-Angaben ...
 //-->
 </STYLE>
 </HEAD>
 <BODY>
 </BODY>
 </HTML>
```

Im Dateikopf einer HTML-Datei kann man mit einem HTML-Befehl des Typs »LINK« eine Datei referenzieren, die Style-Sheet-Angaben enthält. Innerhalb dieses Befehls sollten immer die Angaben REL=stylesheet TYPE="text/css" stehen. Bei der Angabe HREF= wird die gewünschte Datei notiert. Die Angabe muß in Anführungszeichen stehen. Wenn sich die CSS-Datei in einem anderen Verzeichnis oder auf einem anderen Server befindet, muß man mit relativen Pfadangaben oder absoluten URL-Adressen arbeiten.

Bei der referenzierten Datei muß es sich um eine reine Textdatei handeln, die die Endung .css haben sollte. Die Datei sollte nichts anderes als Formatdefinitionen enthalten, also auch keine HTML-Befehle. Eine solche Datei läßt sich mit jedem einfachen Texteditor erstellen

Wenn man mit »LINK« eine Datei mit Style-Sheet-Angaben referenziert, braucht man keinen Bereich <style...></style>. Im Beispiel wird dennoch ein solcher Bereich definiert. Das soll zeigen, daß man beide Arten, Style-Sheet-Angaben zu definieren, durchaus kombinieren kann. Benutzt man beide Arten, haben Formate, die direkt innerhalb von <style...></style> definiert werden, im Konfliktfall Vorrang vor den referenzierten Formaten. So kann man etwa immer wieder verwendete Formate importieren und einige davon mit individuellen Formaten für eine spezielle Seite überschreiben.

Es gibt hierfür auch eine spezielle CSS-Syntax mit dem gleichen Zweck, doch ist die hier aufgeführte Methode mit »LINK« – noch – sicherer, so daß die Alternative hier derzeit unerwähnt bleibt.

Style-Sheet-Dateien für unterschiedliche Ausgabemedien

Bildschirm und Drucker beispielsweise sind sehr unterschiedliche Ausgabemedien für ansprechend gestaltete Daten. Beide haben ihre eigenen Gesetze. Während am Bildschirm etwa helle Schriften auf dunklen Hintergründen attraktiv aussehen, ist das für die Ausgabe am Drucker keine gute Lösung. Bei der Druckerausgabe sehen dagegen Absatzeinzüge von mindestens 2cm besser aus, während durch entsprechende Angaben am Bildschirm möglicherweise kostbarer Präsentationsraum verschenkt wird. Deshalb lassen sich für unterschiedliche Ausgabemedien verschiedene Style-Sheet-Dateien einbinden:

```
<HTML>
 <HEAD>
 <TITLE>Titel der Datei</TITLE>
 <LINK REL=stylesheet MEDIA="screen" HREF="website.css">
 <LINK REL=stylesheet MEDIA="print" HREF="printer.css">
 <LINK REL=stylesheet MEDIA="aural" HREF="speaker.css">
 </HEAD>
 <BODY>
 </BODY>
 </HTML>
```

Um Style-Sheets für verschiedene Ausgabemedien bereitzustellen, notiert man diese in unterschiedlichen Style-Sheet-Angaben in separaten Dateien. Man lege für jedes Ausgabemedium eine Textdatei mit der Endung ».css« an, die nichts anderes als die gewünschten Style-Sheet-Definitionen enthält. In einer HTML-Datei können alle erstellten Style-Sheet-Dateien wie im Beispiel eingebunden werden. Wichtig ist dabei die Angabe »MEDIA=«. Durch diese Angabe wird festgelegt, für welches Ausgabemedium die Datei verwendet werden soll, die bei der Angabe »HREF=« einzubinden ist.

Derzeit sind bei »MEDIA« folgende Angaben bekannt:
media="screen" für Style-Sheets, die bei der Bildschirmpräsentation wirksam sein sollen;
media="print" für Style-Sheets, die bei der Ausgabe über Drucker wirksam sein sollen;
media="aural" für Style-Sheets, die wirksam sein sollen, wenn der Inhalt der HTML-Datei per
Sprachausgabe über Lautsprecher erfolgt;
media="projection" für Style-Sheets, die wirksam sein sollen, wenn der Inhalt der HTML-Datei per Diaprojektor oder Overhead-Projektor erfolgt;
media="braille" für Style-Sheets, die wirksam sein sollen, wenn der Inhalt der HTML-Datei über taktile Braille-Medien erfolgt.
media="tv" für Style-Sheets, die wirksam sein sollen, wenn der Inhalt der HTML-Datei über Fernsehtechnik erfolgt.
media="handheld" für Style-Sheets, die wirksam sein sollen, wenn der Inhalt der HTML-Datei
über Handys, Palmtops oder ähnliche Geräte mit kleinem Display erfolgt.
media="all" für Style-Sheets, die in allen Medientypen wirksam sein sollen.

Bislang interpretiert der MS Internet Explorer diese Angaben ab der Software-Version 4.0 zu einem gewissen Teil. Netscape 4.0 findet zwar die richtige Style-Sheet-Datei für die Bildschirmformate, ignoriert jedoch beispielsweise Style-Sheet-Dateien mit Formaten für den Druck.

Formate für HTML-Tags definieren

Man kann für gewöhnliche HTML-Tags, zum Beispiel für HTML-Tags zu Absatztypen und Textgestaltung oder für Tabellen, mit Hilfe von Style-Sheet-Angaben Formate definieren. Wenn man das entsprechende HTML-Tag dann in der HTML-Datei verwendet, werden alle Formate angewendet, die man für das betreffende HTML-Tag definiert hat. So läßt sich z.B. für Überschriften 1. Ordnung definieren, daß diese in der Schriftart Helvetica, 20 Punkt, rot, fett und kursiv mit einem Absatznachabstand von 16 Punkt angezeigt werden. Wenn man dann im Text der HTML-Datei eine Überschrift 1. Ordnung wie gewohnt definiert, werden die definierten Formateigenschaften bei der Anzeige im Browser berücksichtigt (Datei: »CSS-Formate in Tags.html«).

```
<HTML>
 <HEAD>
 <TITLE>CSS-Formate in Tags.html</TITLE>
 <STYLE TYPE="text/css">
 <!--
 h1 { font-size:48pt; color:#FF0000; font-style:italic; }
 p,li  { font-size:12pt;
     line-height:14pt;
     font-family:Helvetica,Times;
     letter-spacing:0.2mm;
     word-spacing:0.8mm; }
 //-->
 </STYLE>
 </HEAD>
 <BODY>
 <H1>&Uuml;berschrift 1. Ordnung</H1>
 <P>ein normaler Textabsatz</P>
 <UL>
 <LI>Ein Listenpunkt
 <LI>Ein anderer Listenpunkt
 </UL>
 </BODY>
 </HTML>
```

An erlaubten Stellen lassen sich Style-Sheets in HTML einbinden.
Dazu notiert man zuerst das gewünschte Tag, und zwar ohne spitze Klammern. Im Beispiel werden die Tags »H1«, »P« und »LI« notiert. Wenn man gleichartige Formate für mehrere Tags definieren will, gibt man alle gewünschten Tags an und trennt diese durch Kommata, so wie im Beispiel »p,li«.

Es bedeutet also:
h1 { font-family:Helvetica }
h2 { font-family:Helvetica }
dasselbe wie:
h1, h2 { font-family: Helvetica }

Dahinter folgen die dazugehörigen gewünschten Formatdefinitionen, und zwar in geschweiften Klammern { und }. Innerhalb dieser kann man eine oder mehrere Formateigenschaften definieren. Jede Zuweisung besteht aus einem Namen, z.B. »color«, und einem Wert, z.B. »#FF0000«, getrennt durch einen Doppelpunkt. Jede Formatdefinition wird jeweils durch einen Strichpunkt abgeschlossen. Nur bei der letzten vor der abschließenden geschweiften Klammer darf der Strichpunkt entfallen.

Wichtig: Zwar sind innerhalb einer Formatdefinition durchaus Leerzeichen erlaubt, aber der Netscape-Browser ignoriert dann die Angaben. Obwohl es nicht so gut lesbar ist, ist es deshalb vorläufig auf jeden Fall besser, keine Leerzeichen innerhalb einer Formatdefinition zu verwenden.

Im obigen Beispiel wird dem HTML-Tag für Überschriften 1. Ordnung eine Schriftgröße von 48 Punkt (font-size:48pt;), die Schriftfarbe rot (color:#FF0000;) und der Schriftstil kursiv (font-style:italic;) zugewiesen. Wird das betreffende Tag dann innerhalb der Datei verwendet, werden die definierten Formate darauf angewendet.

Style-Sheet-Angaben für Tags wie <P> oder <LI> werden möglicherweise nur interpretiert, wenn ein einleitendes und ein abschließendes Tag notiert ist.

Format-Unterklassen definieren

Man kann mit Hilfe von Style-Sheets Unterklassen von gewöhnlichen HTML-Tags definieren. Das ist vor allem für Tags zu Absatztypen und Textgestaltung oder für Tabellen sinnvoll. So lassen sich beispielsweise vom Tag für Überschriften 1. Ordnung zwei Varianten erzeugen, etwa eine mit schwarzer und eine mit roter Schrift. Dazu vergibt man Namen für die entsprechenden Unterklassen (Datei: »CSS-Unterklassen.html«):

```
<HTML>
  <HEAD>
  <TITLE>CSS-Unterklassen.html</TITLE>
  <STYLE TYPE="text/css">
  <!--
  p.normal { font-size:10pt; color:black; }
  p.gross { font-size:12pt; color:black; }
  p.klein { font-size:8pt; color:black; }
  all.rot { color:red; }
  .blau { color:blue; }
  //-->
  </STYLE>
  </HEAD>
  <BODY>
  <P CLASS="normal">Normaler Textabsatz mit Schrift 10 Punkt schwarz</P>
  <P CLASS="gross">Textabsatz mit Schrift 12 Punkt schwarz</P>
  <P CLASS="klein">Textabsatz mit Schrift 8 Punkt schwarz</P>
  <P CLASS="rot">roter Textabsatz</P>
  <ADDRESS CLASS="rot">roter Absatz für Adressen</ADDRESS>
  <BLOCKQUOTE CLASS="blau">blaues Zitat</BLOCKQUOTE>
  </BODY>
  </HTML>
```

An erlaubten Stellen kann man Style-Sheets in HTML einbinden.
Unterklassen von HTML-Tags bilden man dort, indem man zuerst das Tag notiert, im Beispiel <P>. Dahinter folgt ein Punkt und dahinter ein Name für die Unterklasse.

Anstelle eines bestimmten Tags läßt sich auch das Schlüsselwort »all« notieren. Dadurch definiert man eine Unterklasse mit Formatierungen, die anschließend auf irgendein Tag angewendet werden kann. Es ist auch erlaubt, das Schlüsselwort »all« wegzulassen. In diesem Fall beginnt die Formatdefinition mit einem Punkt. Dahinter folgt der Name der Unterklasse. Im Beispiel wird die Unterklasse blau auf diese Weise definiert.

Die Namen hinter dem Punkt kann man frei vergeben. Sie sollten nicht zu lang sein und keine Leerzeichen und keine deutschen Umlaute enthalten.

Innerhalb des Dateikörpers lassen sich definierte Unterklassen anwenden. Dazu notiert man im einleitenden HTML-Tag die Angabe »CLASS« und dahinter den Namen der Unterklasse, der zuvor vergeben wurde. Im obigen werden zunächst drei Varianten des normalen Fließtext-absatzes <P> notiert. In den beiden Zeilen darunter kann man die Wirkung des Schlüssel-wortes »all« sehen. Verschiedene Tags (im Beispiel <P> und <ADDRESS>) können die gleiche Unterklasse von Formaten benutzen.

Es ist durchaus erlaubt, für mehrere HTML-Tags den gleichen Klassennamen zu vergeben. So kann man z.B. Unterklassen wie »p.gross«, »h1.gross« und »li.gross« definieren.

Um größeren Textabschnitten, die z.B. aus Überschriften, Fließtext und Listen bestehen, bestimmte Formateigenschaften zuzuweisen, definiert man am besten gewünschte Unterklassen für das Tag <DIV></DIV>. So läßt sich bspw. eine Unterklasse »div.rot« {color:red; } definieren. Wenn dann größere Textabschnitte in das DIV-Tag eingeschlossen werden, erhalten alle Absatztypen innerhalb dieses Abschnitts die Textfarbe Rot.
Wenn man mit dem Schlüsselwort »all« arbeitet, können Angaben wie »font-size:10pt« unter Umständen wenig Sinn machen.
Wird etwa »all.Meine« { font-size:10pt;} angeben und im Text dann <H1 CLASS="Meine"> notiert, wird auch die Überschrift 1. Ordnung mit 10 Punkt dargestellt. Sinnvoller ist in diesem Fall das Arbeiten mit prozentualen Angaben, die in Style-Sheets ebenfalls erlaubt sind. So kann man z.B. eine allgemeingültige Unterklasse »all.Meine« {font-size:150%;} definieren. Wird diese Unterklasse dann einem Tag zugewiesen, wird der darin enthaltene Text in 1,5facher Größe dargestellt, und zwar in Bezug auf die sonst übliche Größe.

Formate für verschachtelte HTML-Tags definieren

Wird nichts anderes angeben, übernimmt ein Tag innerhalb eines anderen Tag-Containers dessen Eigenschaften und fügt seine eigenen Eigenschaften nur hinzu. Wenn man etwa für Überschriften 1. Ordnung die Schriftart Times und die Farbe rot definiert, erscheint Text, der innerhalb einer solchen Überschrift mit <I></I> formatiert wird, ebenfalls rot und in Times, aber zusätzlich kursiv.

Man kann jedoch mit Hilfe von Style-Sheets auch bestimmen, daß ein Tag bestimmte Eigenschaften nur dann hat, wenn es innerhalb eines bestimmten anderen HTML-Tags vorkommt. So läßt sich etwa bestimmen, daß <I></I> innerhalb von Überschriften nicht kursiv, sondern in blauer Farbe dargestellt wird, während der gleiche Befehl innerhalb anderer Tags nach wie vor nichts anderes als eine kursive Darstellung bewirkt.

```
<HTML>
 <HEAD>
 <TITLE>Titel der Datei</TITLE>
 <STYLE TYPE="text/css">
 <!--
 h1 { color:red; }
 h1 i { color:blue; font-weight:normal; }
 //-->
 </STYLE>
 </HEAD>
 <BODY>
 <H1>Wir lernen <I>Style-Sheets</I></H1>
 <P>Wir lernen <I>Style-Sheets</I></P>
 </BODY>
 </HTML>
```

An erlaubten Stellen lassen sich Style-Sheets in HTML einbinden.

Im Beispiel wird festgelegt, daß Textabschnitte, die mit dem Tag <I></I> formatiert sind, nicht wie sonst üblich kursiv, sondern normal, stattdessen aber mit blauer Farbe dargestellt werden. Aber nur dann, wenn dieser Textbereich innerhalb einer Überschrift 1. Ordnung

vorkommt. Dazu notiert man zuerst das übergeordnete Tag, im Beispiel »H1«, und dahinter, durch ein Leerzeichen getrennt, das untergeordnete Tag, im Beispiel »I«.

In der ersten Zeile innerhalb des <BODY>-Bereichs im Beispiel kommt diese spezielle Definition zum Tragen. In der zweiten Zeile dagegen, wo <I></I> innerhalb eines anderen Tags vorkommt, hat es die übliche Wirkung.

Unabhängige Formate definieren

Man kann Formate vordefinieren, die sich anschließend auf beliebige geeignete HTML-Tags anwenden lassen. So z.B. ein unabhängiges Format mit der Eigenschaft »fett/kursiv«. Dieses unabhängige Format kann man dann innerhalb von Tags zusätzlich zu anderen Formaten anwenden, die für das betreffende Tag definiert sind (Datei: »CSS-unabh-Formate.html«).

```
<HTML>
 <HEAD>
 <TITLE>CSS-unabh-Formate.html</TITLE>
 <STYLE TYPE="text/css">
 <!--
 p,li,dd,dt,blockquote { color:red;font-family:Times; margin-top:1cm; margin-
left:1cm; }
 #fettkursiv { font-weight:bold; font-style:italic; }
 //-->
 </STYLE>
 </HEAD>
 <BODY>
 <P ID="fettkursiv">Extra-Formatierung</P>
 <P>Das ist formatierter Text mit <EM ID="fettkursiv">Extra-
Formatierung</EM></P>
 </BODY>
 </HTML>
```

Im Beispiel wird für Fließtext-Standard-Tags wie <P>, <LI>, <DT>, <DD> und <BLOCKQUOTE> ein einheitliches Format mit verschiedenen Angaben definiert. Ferner ein unabhängiges Format mit dem Namen »fettkursiv«. Namen von unabhängigen Formaten müssen ein Gatterzeichen # vorangestellt bekommen. Für die Namen gelten die gleichen Regeln wie für Namen von Unterklassen. Im Beispiel werden dem unabhängigen »fettkursiv« die beiden Style-Sheet-Angaben zugewiesen, die einen Text fett und kursiv machen.

Im Dateikörper kann man freie Formate innerhalb von Tags anwenden. Dazu notiert man im einleitenden Tag die Angabe »ID=« und dahinter, in Anführungszeichen, den Namen des unabhängigen Formats ohne Gatterzeichen. Dadurch werden dem Inhalt die Eigenschaften des definierten unabhängigen Formats zugewiesen. Im Beispiel wird »fettkursiv« einmal innerhalb eines einleitenden <P>-Tags und einmal innerhalb eines einleitenden <EM>-Tags angegeben.
Daraus läßt sich ersehen, wie unabhängige Formate einsetzbar sind und wie sie andere Formate ergänzen.

Unabhängige Formate wirken normalerweise additiv, d.h. sie übernehmen das vorhandene Format und fügen ihre eigenen Eigenschaften lediglich hinzu. Sie haben jedoch im Konfliktfall Vorrang vor anderen Formatierungen: Wenn man z.B. für Überschriften 1. Ordnung eine rote Farbe definiert und einer solchen Überschrift ein unabhängiges Format zuweist, in dem die Farbe Blau definiert wird, dann wird die Überschrift blau.

Allgemeines zur Schriftformatierung (bei CSS)

Unter Schriftformatierung sind Angaben zu Schriftarten, Schriftgrößen, Schriftfarben, Schriftgewicht, Zeichen- und Wortabständen usw. zu verstehen. Sinnvoll sind solche Angaben für alle HTML-Tags, die Text enthalten können. Dazu gehören ebenfalls Tags für Tabellen. Auch auf das <BODY>-Tag lassen sich die hier aufgelisteten Style-Sheet-Angaben anwenden – dann gelten die Angaben für alle Textelemente der gesamten HTML-Datei.

Schriftart (font-family)

Unter Schriftarten sind Schriftarten wie etwa Arial, Helvetica, Times Roman usw. zu verstehen. Auch Schriftfamilien wie Sans Serif usw. gehören dazu.

Mit dem hier beschriebenen Befehl kann man Schriftarten angeben, ohne sich darum zu kümmern, ob und wie die Schriftart beim Anwender angezeigt werden kann. Falls die angegebene Schriftart nicht angezeigt werden kann, bleibt die Angabe wirkungslos.
Es gibt zwar eine Möglichkeit, Schriftarten durch Angabe einer bestimmten Schriftarten-Datenquelle zu erzwingen, doch ist sie (a) noch nicht ausgereift und (b) so komplex, daß sie hier nicht erläutert werden kann.

Beispiel (Style-Sheet-Definition im Dateikopf):

```
<STYLE TYPE="text/css">
  h1,h2,h3 { font-familiy:Avantgarde,Arial }
</STYLE>
```

Mit »font-family:« kann man eine oder mehrere Schriftarten bestimmen. Bei mehreren angegebenen Schriftarten ist die Reihenfolge der Angabe entscheidend: Ist die erste angegebene Schriftart verfügbar, wird diese verwendet. Ist sie nicht verfügbar, wird die zweite Schriftart verwendet, falls diese verfügbar ist, usw. Angaben zu Schriftarten, die nicht angezeigt werden können, werden vom Browser ignoriert er erkennt eine Typenzugehörigkeit und verwendet eine ähnliche Schriftart.
Trennen Sie die Schriftartennamen durch Kommata. Das W3-Konsortium empfiehlt, Schriftartnamen, die Leerzeichen enthalten, in Anführungszeichen zu setzen, also z.B. »font-familiy:"Century Schoolbook",Times«.
Folgende Schriftfamilien sind fest vordefiniert – diese Angaben kann man also neben Schriftartnamen benutzen: »serif«, »sans-serif«, »cursive«, »fantasy« und »monospace«.

Schriftstil (font-style)

Schriftstil, besser »Schriftschnitt«, bedeutet hier die Neigung der Schrift.

Beispiel (Style-Sheet-Definition für HTML-Tag im Text):

```
<P STYLE="font-style:italic">Text</P>
```

Mit »font-style:« läßt sich der Schriftstil bestimmen. Folgende Angaben sind möglich:

italic = Schriftstil kursiv
oblique = Schriftstil kursiv
normal = normaler Schriftstil

Schriftvariante (font-variant)

Als besondere Schriftvariante stehen Kapitälchen zur Verfügung.

Beispiel (Style-Sheet-Definition im Dateikopf):

```
<STYLE TYPE="text/css">
  h4 { font-variant:small-caps }
</STYLE>
```

Mit »font-variant:« kann man die Schriftvariante bestimmen. Folgende Angaben sind möglich:

small-caps = Kapitälchen
normal = normale Schriftvariante

Schriftgröße (font-size)

Schriftgröße ist die Darstellungsgröße der Schrift in Punkt.

Beispiel (Style-Sheet-Definition für HTML-Tag im Text):

```
<H2 STYLE="font-size:24pt">Text</H2>
<P STYLE="font-size:130%">Text</P>
```

Mit »font-size:« läßt sich der Schriftgrad bestimmen. Erlaubt ist eine numerische Angabe. Auch eine prozentuale Angabe ist möglich. Die Angabe 130% im Beispiel bedeutet: 130% im Verhältnis zur normalen Schriftgröße, die als 100% gedacht ist.

Alternativ zu numerischen Angaben sind auch folgende »ungenauen« Angaben möglich:

xx-small = winzig
x-small = sehr klein
small = klein
medium = mittel
large = groß
x-large = sehr groß
xx-large = riesig
smaller = sichtbar kleiner als normal
larger = sichtbar größer als normal

Man kann die Angabe zur Schriftgröße mit der Angabe zur Zeilenhöhe kombinieren, indem man beide Angaben innerhalb der Angabe »font:« mit der folgenden Syntax notiert (Beispiel): p { font:12pt/14pt }. Hier ist 12pt die Schriftgröße und 14pt die Zeilenhöhe.

Schriftgewicht (font-weight)

Schriftgewicht ist der »Fettigkeitsgrad«, in dem die Schrift dargestellt wird.

Beispiel (Style-Sheet-Definition im Dateikopf):

```
<STYLE TYPE="text/css">
   dt { font-weight:bold }
   em { font-weight:600 }
 </STYLE>
```

Mit »font-weight:« kann man das Schriftgewicht bestimmen. Folgende Angaben sind möglich:

bold = fett
bolder = extra-fett
lighter = dünner
100,200,300,400,500,600,700,800,900 = extra-dünn (100) bis extra-fett (900)
normal = normales Schriftgewicht

Bei den numerischen Werten entspricht die Angabe 500 dem im DTP-Bereich üblichen Begriff »medium«, die Angabe 700 dem Begriff »bold«.
Wohl kaum eine installierte Schriftart unterstützt jedoch alle erlaubten Angaben zum Schriftgewicht.

Schrift allgemein (font)

Diese Angabe ist eine Zusammenfassung der folgenden fakultativen Einzelangaben:
- font-family
- font-style
- font-variant
- font-size
- font-weight
- line-height

Beispiel (Style-Sheet-Definition für HTML-Tag im Text):

```
<P STYLE="font:bold italic 12pt Palatino, serif">Text</P>
```

Mit »font:« lassen sich verschiedene Schriftformatierungen mischen. Erlaubt sind die üblichen Wertangaben zu den erlaubten Stlye-Sheet-Angaben, die in »font:« zusammengefaßt sind. Die Reihenfolge der Angaben ist egal.

Wortabstand (word-spacing)

Mit dieser Angabe wird der Abstand zwischen Wörtern (Wortpause) im Text bestimmt.

Beispiel (Style-Sheet-Definition im Dateikopf):

```
h3 { word-spacing:8mm }
```

Mit »word-spacing:« kann man die Wortpausen bestimmen. Erlaubt ist eine numerische oder eine prozentuale Angabe.
Diese Angabe wird von Netscape und dem MS Internet Explorer in den Versionen 4.x noch nicht interpretiert.

Zeichenabstand (letter-spacing)

Mit dieser Angabe kann man den Abstand zwischen den Buchstaben bzw. Zeichen im Text (Räume) bestimmen.

Beispiel (Style-Sheet-Definition für HTML-Tag im Text):

```
<B STYLE="letter-spacing:3em">fetter breiter Text</B>
```

Mit »letter-spacing:« lassen sich die Räume bestimmen. Erlaubt ist eine numerische oder eine prozentuale Angabe.
Diese Angabe wird nur vom Explorer, nicht von Netscape interpretiert. Der Explorer erlaubt keine Prozentangaben.

Textdekoration (text-decoration)

Hiermit lassen sich bestimmte Effekte/Auftritte des Textes bestimmen.

Beispiel (Style-Sheet-Definition im Dateikopf):

```
strong { text-decoration:underline }
```

Mit »text-decoration:« sind folgende Angaben möglich:

underline = unterstrichen
overline = überstrichen
line-through = durchgestrichen
blink = blinkend
none = normal (keine Text-Dekoration)

Der Explorer interpretiert die Angabe »blink« nicht, Netscape 4.x die Angabe »overline« nicht.

Texttransformation (text-transform)

Mit dieser Angabe kann in einem Textbereich Klein- oder Großbuchstaben oder Kapitälchen erzwingen, unabhängig davon, wie die einzelnen Buchstaben tatsächlich in der Datei stehen.

Beispiel (Style-Sheet-Definition für HTML-Tag im Text):

```
<H1 STYLE="text-transform:uppercase">alles grossgeschrieben</H1>
```

Mit »text-transform:« wird eine Texttransformation erzwungen. Folgende Angaben sind möglich:

capitalize = Wortanfänge als Großbuchstaben
uppercase = Nur Großbuchstaben
lowercase = Nur Kleinbuchstaben
none = normal (keine Text-Transformation)

Der MS Internet Explorer 4.x interpretiert die Angabe »capitalize« noch nicht.

Textfarbe (color)

Mit dieser Angabe können Sie Textvordergrundfarben bestimmen.

Beispiel (Style-Sheet-Definition im Dateikopf):

```
b,i { color:#CC0000 }
```

Mit »color:« kann man die Textfarbe bestimmen. Erlaubt sind Farbangaben.

Textschatten (text-shadow)

Hiermit kann für Text ein Schatteneffekt erzeugt werden.

Beispiel (Style-Sheet-Definition für HTML-Tag im Text):

```
<P STYLE="text-shadow:black; font-size:24pt;">Text</P>
```

Mit »text-shadow:« kann man einen Textschatten erzwingen. Erlaubt sind Farbangaben oder der Wert »none« für »keinen Textschatten«.
Dieser Befehl, der zur Version 2.0 der CSS-Style-Sheets gehört, wird von Netscape und Explorer in den Versionen 4.x noch nicht interpretiert.

Kategorien Mime-Typen

Die Abkürzung MIME steht für »Multipurpose Internet Mail Extensions«. MIME-Typen sind ein Internet-Standard, um Dateitypen anzugeben. Im Zusammenhang mit multimedialen Elementen auf WWW-Seiten werden diese Angaben in Zukunft immer wichtiger.

MIME-Typen werden bei der Kommunikation zwischen WWW-Server und WWW-Browser eingesetzt. Sowohl der WWW-Server als auch der WWW-Browser unterhält eine Liste mit ihm bekannten Dateitypen. In vielen WWW-Browsern (z.B. bei Netscape) ist das die Liste der
sogenannten »Helper Applications«. Beim Übertragen vom Server zum Browser wird über das HTTP-Protokoll der MIME-Type mit übertragen. Aufgrund seiner Liste mit MIME-Typen weiß der WWW-Browser, wie er die Datei zu behandeln hat.

Zwar unterszützen viele Plattformen wie zum Beispiel Windows Standardverknüpfungen zwischen Dateinamenerweiterungen und Dateitypen. Doch ist diese Verknüpfung nicht eindeutig genug. So kann die Dateinamenerweiterung .doc beispielsweise ein Word-Dokument, ein FrameMaker-Dokument oder eine einfache Textdatei bedeuten. Das System der Mime-Typen hat die Aufgabe, solche Zweideutigkeiten zu beseitigen und den beteiligten Programmen (WWW-Server-Software, WWW-Browser) ein eindeutiges Identifizierungsschema für den Datentyp von Dateien bereitzustellen.

MIME-Typen werden nach folgendem Schema angegeben:

Kategorie/Unterkategorie

sind z.B. »text«, »image« oder »audio«. Unterkategorien von »text« sind etwa »plain« (Datei ist eine reine Textdatei) oder »html« (Datei ist eine HTML-Datei). Unterkategorien von »image« ist z.B. »gif« (Datei ist eine Grafik im GIF-Format).

wichtiger Mime-Typen

Mime-Type	Dateinamen-Erweiterung(en)	Beschreibung
application/acad (NCSA)	dwg	AutoCAD-Dateien
application/dxf (CERN)	dxf	AutoCAD-Dateien
application/mif	mif	Maker Interchange Format (Adobe FrameMaker)
application/msword	doc dot	MS-Word-Dateien
application/mspowerpoint	ppt ppz pps pot	MS-Powerpoint-Dateien
application/msexcel	xls xla	MS-Excel-Dateien
application/mshelp	hlp chm	MS-Windows-Hilfe-Dateien
application/octet-stream	com exe bin dll class	Ausführbare Dateien bzw. Programmcode-Dateien

application/pdf	pdf	PDF-Dateien (Adobe Acrobat Exchange/Reader)
application/postscript	ai eps ps	Postscript-Dateien (Adobe)
application/rtf	rtf	RTF-Dateien (Microsoft)
application/x-sh	sh	Bourne Shell Script (Unix)
application/x-csh	csh	C Shell Script (Unix)
application/x-latex	latex	LaTeX-Quelldatei (Unix)
application/x-mif	mif	Maker Interchange Format (Adobe FrameMaker Unix)
application/x-tar	tar	tar-Archivdatei (Unix)
application/x-cpio	bcpio	CPIO-Datei alt binär (Unix)
application/x-bcpio	cpio	CPIO-Datei (Posix)
application/x-sv4cpio	sv4cpio	CPIO-Datei (SVR4)
application/x-sv4crc	sv4crc	CPIO-Datei (SVR4 mit CRC)
application/x-hdf	hdf	NCSA HDF Data File
application/x-ustar	ustar	tar-Archivdatei (Posix)
application/x-shar	shar	Shell-Archiv-Datei (Unix)
application/x-tcl	tcl	TCL-Script (Unix)
application/x-dvi	dvi	TeX dvi (Unix)
application/x-texinfo	texinfo texi	Emacs Texinfo (Unix)
application/x-troff	t tr roff	troff-Dateien (Unix)
application/x-troff-man	man	troff mit MAN-Makros (Unix)
application/x-troff-me	me	troff mit ME-Makros (Unix)
application/x-troff-ms	ms	troff mit MS-Makros (Unix)
application/x-netcdf	nc cdf	Unidata netCDF (Unix)
application/x-wais-source	src	WAIS-Quelldatei (Unix)
application/x-www-form-urlencoded		HTML-Formulardaten an CGI
audio/basic	au snd	AU- und SND-Sound-Dateien
audio/x-aiff	aif aiff aifc	AIFF-Sound-Dateien
audio/x-dspeeh	dus cht	Sprach-Dateien
audio/x-midi	midi mid	MIDI-Dateien
audio/x-pn-realaudio	ram ra	RealAudio-Dateien
audio/x-pn-realaudio-plugin	rpm	RealAudio-Plugin-Dateien

image/cmu-raster	ras	CMU-Raster
image/x-freehand	fh4 fh5 fhc	Freehand-Grafik
image/gif	gif	GIF-Grafik
image/ief	ief	Image Exchange Format
image/jpeg	jpeg jpg jpe	JPEG-Grafik
image/x-portable-anymap	pnm	PBM Anymap-Datei
image/x-portable-bitmap	pbm	PBM Bitmap-Datei
image/x-portable-graymap	pgm	PBM Graymap-Datei
image/x-portable-pixmap	ppm	PBM Pixmap-Datei
image/x-rgb	rgb	RBG-Grafik
image/x-windowdump	xwd	X-Windows Dump
image/tiff	tiff tif	TIFF-Grafik
text/css	css	CSS-Style-Sheet-Datei
text/html	html htm	HTML-Datei
text/javascript	js	JavaScript-Datei
text/plain	txt c cc g h hh m f90	reine Text-Datei
text/richtext	rtx	MIME Richtext
text/tab-separated-values	tsv	Datentextdatei mit Tabulatoren als Feldtrenner
text/x-setext	etx	Struct.erw. Text
text/x-sgml	sgm sgml	SGML-Datei
video/mpeg	mpeg mpg mpe	MPEG Video
video/quicktime	qt mov	Quicktime-Video
video/x-msvideo	avi	Microsoft AVI-Video
video/x-sgi-movie	movie	Microsoft SGI-Video
x-world/x-vrml	wrl	VRML-Dateien

17.10.1999
Dr. Michael Meissner
(Dank an S. Münz)