Die Auszeichnungssprache im Web-Publishing

1. Einführung

Sicher fällt Ihnen sofort auf, dass im Titel nichts von HTML steht. Da aber HTML weder der Anfangs- noch der Endpunkt einer langen Entwicklung darstellt, ist es hier bewusst weggelassen worden.

Um es gleich vorwegzunehmen: HTML und XHTML stellen **kein** WYSIWYG-(What you see is what you get)-Textverarbeitungssystem dar. Wie auch im Fall des Drucksatz-Systems TEX wird hier eine strikte Trennung zwischen Textinhalten und Layout vorgenommen, auch erfolgt die Formatierung erst zu einem späteren Zeitpunkt.[1] Dies bedeutet, dass der Autor nur seine Inhalte und die Art von Containern bestimmt, in die der Text eingebettet wird. Die Darstellung wird vom Browser oder einem anderweitigen Ausgabemedium vorgenommen. Das bedeutet aber nicht, dass es gar keine Möglichkeiten der Einflussnahme auf die Gestaltung gibt, Stilanweisungen werden aber separat notiert und sind austauschbar. Die scheinbar umständliche Vorgehensweise besitzt aber ihre Vorteile: Das Layout kann vom interpretierenden Ausgabegerät oder vom Lesenden an spezielle Bedingungen angepasst werden.

Als die erste Version der Hypertext Markup Language (**HTML**)[2] von Tim Barners-Lee entwickelt wurde, stand als Ziel der einfache und plattformübergreifende Dokumentenaustausch wissenschaftlicher Dokumente über das Internet im Vordergrund. Dazu sollte keine komplizierte Sprache erlernt werden müssen.

Das entwickelte HTML stützt sich dabei auf bereits bestehende Philosophien, nämlich der Standard Generalized Markup Language (**SGML**), die auf frühen Entwicklungen von IBM basiert.[3] Diese Auszeichnungssprache besteht aus so genannten Tags[4], mit denen bestimmte logische oder physische Strukturelemente (Container) eines Dokuments eindeutig und rechner- und somit auch hardware-unabhängig bezeichnet werden können.[5] Das Dokument wird als einfacher Text im ASCII- bzw. ANSI-Standard (also nicht binärkodiert) erfasst, d.h. in einem ebenfalls plattformunabhängigen Textformat.

---

[1] Diese Vorgehensweise wird auch in modernen Textverarbeitungssystemen (Stilvorlagen, Formatvorlagen) angeboten, leider von den Autoren aber viel zu selten benutzt.

[2] Zu unterscheiden von HTTP (Hypertext Transfer Protocol), dem im Internet verwendeten Übertragungsprotokoll von HTML-Dateien.

[3] Von der International Standardization Organization (ISO) unter ISO 8879 standardisiert.

[4] *engl.*, Markierung, Etikett.

[5] Eine ähnliche Philosophie wird auch in der Textverarbeitungssoftware WordPerfect vollzogen: Im Steuerzeichenfenster kann man sich den Text hinter der WYSIWYG-Fassade ansehen und auch editieren.

Ein Beispiel für ein solche Tags wären z.B.

- `<überschrift>`Das ist eine Überschrift`</überschrift>`
- `<absatz>`Ein ganz langer ... Absatz`</absatz>`.

Wie an obigen Beispiel deutlich wird, wird also eine in sich geschlossene Texteinheit durch ein einleitendes Tag begonnen und mit einem abschließenden Tag beendet.

Die im Beispiel vorgeschlagenen Tags könnten dann von einem SGML-Interpreter interpretiert werden, der dann für die Darstellung eigene oder fremde Stildefinitionen verwendet. SGML selbst ist aber noch keine fertige Auszeichnungssprache, sondern nur ein „grammatikalisches" Regelwerk, wie dies geschehen sollte.

HTML ist die bekannteste Form der Realisierung des SGML-Regelwerks.

Als spezielle Entwicklung für die Verwendung in Arbeitsgruppen, Intranets und im Internet ist eine vereinfachte Form des SGML entwickelt worden, nämlich die Extensible Markup Language (XML). Zusätzlich zur Möglichkeit des HTML, die *Formatierung* und Interaktionsmöglichkeiten für den Nutzer zu definieren, ist es im XML auch möglich, die *Bedeutung* von Textinhalten zu definieren. Hierzu ein kleines Beispiel: Sie notieren ein Literaturzitat mit Autor, Titel und weiteren bibliographischen Angaben. Im HTML besteht hierzu nur die Möglichkeit, die einzelnen Bestandteile durch unterschiedliche Formatierung untereinander unterscheidbar zu machen. Im XML können Sie zusätzlich die Bedeutung angeben: z.B. `<autor>`Rainer Zufall `</autor>`. Das Neue daran ist nun, dass Sie als Nutznießer nach derartigen Elementen gezielt suchen können, Informationsbestandteile interaktiv austauschen, verstecken oder neu formatieren können. Diese Funktionalität hängt eng mit einer weiteren neuen Eigenschaft zusammen: nämlich der Erweiterbarkeit. Sie können beliebige neue Tags definieren, ohne dass sie der Interpreter bereits kennen muss.

Für spezielle Anwendungsfälle gibt es bereits Implementationen, d.h. ein oder mehrere Sätze von vordefinierten Stilinformationen. Damit nun ein Interpreter weiß, dass nun ein Word-Dokument oder eine WAP[6]-Nachricht folgt, muss zwingend als erste Angabe der Dokument-Typ (DTD Document Typ Definition) angegeben werden. Und damit wird auch das Anliegen klar: Ein Präsentationsgerät soll in der Lage sein, Dokumente unterschiedlicher Zweckbestimmung darstellen zu können.

[6] Wireless Application Protocol, Protokoll bzw. Anwendung für textbasierte Informationsübertragung für mobile Telefone (Handys).

Speziell für das Internet wird es eine Neuformulierung des HTML 4.0 in XML geben: XHTML. D.h., das zukünftige Web-Publishing wird im XML-Format unter Verwendung der bereits bekannten Tags erfolgen, d.h. die Dokumenttypdefinitionen des HTML 4.0 werden als Definition des XHTML mit nur wenigen XML-typischen Modifikationen übernommen.[7] Für Ihre Arbeit können Sie sich merken: Die Mechanismen von HTML und XHTML sind verschieden; das Aussehen der Textdatei ist gleich.

Am 26. Januar 2000 wurde der XHTML-Standard als Empfehlung vom World Wide Web-Konsortium[8] verabschiedet. Sicher wird es noch einige Zeit dauern, bis sich der Standard durchsetzt, allerdings kann man bereits heute Vorkehrungen treffen, um den Umstellungsaufwand auf ein Minimum zu reduzieren. Internetbrowser der 4. Generation (Netscape 4.x, Internet Explorer 4 und 5.x) kommen damit zurecht; der 2000 erscheinende Netscape-Browser 6 wird XML und XHTML bereits uneingeschränkt beherrschen.

[7] Siehe auch Merkblatt: Wichtige Regeln für die Erstellung von HTML- und XHTML-Dateien.

[8] Siehe auch: http://www.w3.org/TR/xhtml1/

2. HTML-Versionen

2.1. HTML 1.0

- Literatur:
  - Berners-Lee, Tim; Connolly, Daniel: Hypertext Markup Language (HTML) http://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt

Gewissermaßen der Urahn aller Standards. Er definierte die Standardelemente für Überschriften, Textabsätze, für Grafiken und Verweise. Dieser Standard ist heute nicht mehr gebräuchlich.

Zu dieser Zeit beherrschten die Browser Mosaic und Netscape 1.1 bereits Tabellen, die noch nicht Inhalt des ersten Standards waren.

2.2. HTML 2.0 (RFC1866)

- Literatur:
  - W3-Konsortium: Public Text of the HTML 2.0 Specification. http://www.w3.org/MarkUp/html-spec/html-pubtext.html

Dieser Standard wurde im November 1995 verabschiedet. Er gilt heute als kleinster gemeinsamer Nenner aller Browser-Implementationen, obwohl die damaligen Browser bereits einen großeren Tag-Umfang interpretieren konnten (u.a. Frames und JavaScript). Allerdings beherrscht wohl kein Browser das `<link>`-Tag.

Neu sind u.a. Formulare mit ihren Elementen und die Möglichkeit von `<meta>`-Tag-Angaben.

2.3. HTML 3.2

- Literatur:
  - W3-Konsortium: HTML 3.2 Reference Specification vom 14. Januar 1997. http://www.w3.org/TR/REC-html32.html

Ja, dies ist richtig: 3.2 und nicht 3.0. Es war schon eine schwierige Geburt mit dem neuen Standard. Unbeeindruckt vom Erfolg des Netscape-Navigators wurde ein Standard vorbei an den Realitäten entwickelt; der Standard 3.0 sollte aber nie das Licht der Welt erblicken. Das Ergebnis einer verstärkten Zusammenarbeit zwischen W3-Konsortium und kommerziellen

Herstellern wurde dann der Standard HTML 3.2. Neu sind u.a. Tabellen und Befehle zur physischen Textauszeichnung, Frames wurden immer noch nicht standardisiert.

Ein wohl sehr nützliches Tag konnte sich nicht durchsetzen: das `<math>`-Tag. In einer dem Satzsystem TEX ähnlichen Weise sollte die Einbindung mathematischer Formeln und somit auch griechischer Buchstaben erfolgen. Der Wunsch nach der Darstellung mathematischer Ausdrücke wird nun erst im Jahr 2000 Realität mit *MathML*.

2.4. HTML 4.0

- Literatur:
  - W3-Konsortium: HTML 4.01 Specification vom 24. Dezember 1999. http://www.w3.org/TR/REC-html40/

HTML 4.0 wurde am 18.2.1998 als Sprachstandard verabschiedet, als Revision 4.01 liegt er seit Weihnachten 1999 vor (mit neuen Style-Sheet-Definitionen).

Erstmals gelingt es dem W3-Konsortium, wieder einen verbindlichen Standard vorzulegen und nicht den Realitäten hinterher zu rennen – und sich somit wieder als unabhängiges und anerkanntes Standardisierungs-Gremium zu präsentieren.

Neu sind im Standard 4.0 u.a. Frames und Cascading Style Sheets.

2.5. XHTML 1.0

- Literatur:
  - W3-Konsortium: XHTML™ 1.0: The Extensible HyperText Markup Language. Empfehlung vom 26. Januar 2000. http://www.w3.org/TR/xhtml1/

Wie bereits erläutert, werden gegenwärtig verschiedene Dokumenttypen auf Basis der eXtensible Markup Language (XML) entwickelt. Dabei stellt XHTML eine Neuformulierung des HTML 4.0-Standards dar.

3. Elemente der Seitenbeschreibungssprache HTML

3.1. Tags

Tags (Einzahl: Tag[9]) sind Auszeichnungsbefehle für eine Seitenbeschreibungssprache. Tags definieren Formatierungen, aber auch Kennzeichnungen bestimmter Elemente wie z.B. Überschriften. In der Anzeige bleiben Tags unsichtbar, sie sind nur im Quelltext sichtbar.

Tag-Bezeichner und deren Attribute werden im HTML und XML zwischen spitzen Klammern („<“ bzw. „>“) notiert. Textbausteine werden in der Regel zwischen einem einleitendem und einem schließenden Tag eingeschlossen. Das schließende Tag besitzt denselben Namen, wird aber durch ein einleitenden Schrägstrich („/“) kenntlich gemacht. Nur wenige Tags besitzen *kein* schließendes Tag (z.B. `<img>`, `<br>`), im Rahmen des XHTML 1.0 müssen diese aber ebenfalls mit einem schießenden Tag versehen werden (siehe Kapitel wichtige Syntax-Regeln).

Die Attribute eines Tags werden untereinander mit Leerräumen getrennt im einleitenden Tag genannt. Im schließenden Tag erfolgt niemals eine Angabe von Attributen. Sie als Autor brauchen nur diejenigen Attribute anzugeben, die Sie modifizieren möchten und die anderweitig (z.B. über Cascading Style Sheets nicht definiert sind).

Beispiele:

- `<a href="datei.ext">Dies ist ein Link</a>`
- `<img scr="datei.ext" width="150" height="50" alt="Ein Bild" />`

Tags oder Attribute, die einem Browser nicht bekannt sind, werden nicht ausgeführt, und es erfolgt keine Fehlermeldung. Der zwischen einleitendem und schließendem Tag eingeschlossene Text wird aber immer ausgegeben. Der oben angegebene Schrägstrich `... />` ist eine Notwendigkeit von XHTML, da hier jedes Tag abgeschlossen sein muss. Ältere Browser fassen ihn als Attribut, das sie nicht kennen, auf und führen es auch nicht aus.

Es ist erlaubt, Tags mit ihren Attributen über mehrere Zeilen zu notieren. Es ist aber durchaus legitim, auf Zeilenumbrüche gänzlich zu verzichten.

Achten Sie bitte auch eine fehlerfreie Schachtelung (Nesting) von Tags: d.h. das erste einleitende Tag wird mit dem letzten schließenden Tag kombiniert.

---

[9] *engl.* Markierung, Etikett.

Beispiel:

- `<b><i>Das ist ein fetter kursiver Text.</i></b>`

Im vorliegenden Material wird nur noch auch Tags und Attribute eingegangen, die im HTML 4.0-Standard definiert sind. Dabei wird zwischen unbedingt notwendigen und fakultativ angebbaren Attributen unterschieden.

Achten Sie im Weiteren darauf, Tag- und Attribut-Bezeichner in Kleinbuchstaben zu notieren. Die Werte von Attributen müssen unbedingt in Anführungszeichen eingeschlossen werden: Beispiel: `height="520"`. Auch wenn dies die HTML-Standards nicht dringend fordern, wird dies beim XHTML zur Forderung. Wenn man sich von vornherein darauf einstellt, erspart man sich später viel Nacharbeit.

3.2. Attribute

3.2.1. Zweck

Zu weiteren Spezifikation von Tags können dem einleitenden Tag so genannte Attribute beigegeben werden, die aus einem Attributnamen, gefolgt von einem Gleichheitszeichen und dem Attributwert in Apostrophen, bestehen.

Im Folgenden sollen einige wichtige, allgemein einsetzbare Attribute vorgestellt werden.

3.2.2. Universalattribute

Eine Reihe von Attributen lässt sich bei fast allen Tags verwenden. Sie werden hier jetzt kurz vorgestellt.

Universalattribute können für (fast) alle HTML-Tags verwendet werden. Deren Verwendung ist aber nicht obligatorisch.

Bezeichner

- Universalattribut Bezeichner: *id* (von englisch identifier).[10]

[10] In früheren Versionen von HTML wurde statt `<tag-bezeichnung id="Bezeichnung">` das Universalattribut *name* `<tag-bezeichnung name="Bezeichnung">`, das seit der Version HTML 2.0 bekannt war. Es kann nötig sein, dass in einer Übergangszeit sowohl das Attribut *name* als auch das Attribut *id* für

`<tag-bezeichnung id="Bezeichnung">`
Namensangabe für das lokal notierte Tag. Es kann als Zielanker für einen Link oder zum Zweck dynamischer Veränderungen durch ein Skript-Aufruf verwendet werden. Zum Weiteren ist die Nutzung im Rahmen der so genannten Cascading Style Sheets möglich, wenn Formatvorlagen über *ID selectors* spezifiziert werden.

Dieser Name eines Elements gilt dateiweit und muss in der Datei eindeutig sein, d.h., die Bezeichnung darf nicht mehrfach vergeben werden.

In beiden Fällen gilt: Vergeben Sie keine zu langen Namen, erlaubt sind im Wesentlichen nur Buchstaben, Zahlen und der Unterstrich („_“), verwenden Sie keine Leerräume, Umlaute und das ß. Beachten Sie, dass bei Bezügen Groß- und Kleinschreibung unterschieden werden.

Kommentar im Tag

- Kommentar *title*:
  `<tag-bezeichnung title="Das muss ich noch aendern">`

Mit diesem Attribut können Sie jedes Tag mit einem erläuternden Kommentar versehen. Es ist denkbar, dass dieser Kommentar in einem Browser in einem kleinen Tool-Tip-Fenster angezeigt wird, wenn der Anwender über den Bereich mit der Maus verweilt.

Formatanweisungen

- Formatklassenangabe *class*:
  `<tag-bezeichnung class="name">`

Das Attribut *class* gibt an, dass das Tag einer bestimmten Style-Sheet-Klasse angehört.

- Formatanweisung *style*:
  `<tag-bezeichnung style="Formatanweisungen">`

Unter Formatanweisungen können Angaben notiert werden, wie sie im Rahmen von Cascading Style Sheet-Definitionen benutzt werden. Diese Definitionen werden im Abschnitt Cascading Style Sheet behandelt.

---

den Zweck der Cross-Browser-Kompatibilität eingetragen werden müssen.

Mit der Möglichkeit der Angabe von Schriftangaben verschwindet etwas der Unterschied zwischen logischer und physischer Textauszeichnung: Alle Tags können in einer Formatvorlage neu- und umformatiert werden, ein als Fett gekennzeichneter Text muss so gar nicht fett dargestellt werden.

Sprachauszeichnungen

- Sprache *lang* (von englisch language):
  `<tag-bezeichnung lang="language">`
  Als Sprachangabe erfolgt hier ein zweistelliges Sprachkürzel, z.B. `de` für deutsch, `en` für englisch, `fr` für französisch, `it` für italienisch usw. Die Form der Vereinbarung in einem HTML-Dokument sind in der RFC 1766 (Tags for the Identification of Languages) definiert, die Sprachen-Codes sind in der ISO 639-1 festgelegt. Dem primären „Tag“ darf nach einem Bindestrich noch ein „Subtag“ in Form einer Länderinformation, wie z.B. `en-us`, einer Dialektinformation, wie z.B. `no-nynorsk`, oder einer Schreibvariante, wie z.B. `az-arabic`, `az-cyrillic`, folgen. Die Länder-Codes sind in der ISO 3166-1 definiert.
  Gleichartige Definition für XHTML-Dokumente:
  `<tag-bezeichnung xml:lang="language">`
- Textrichtung *dir* (von englisch direction):
  `<tag-bezeichnung dir="richtung">`
  Mögliche Werte für das *dir*-Attribut sind `ltr` (von links nach rechts [left to right]) und `rtl` (von rechts nach links [right to left]). Normalerweise sollte der Browser die Schreibrichtung einer Sprache beherrschen), sicherheitshalber kann man für Sprachen wie Arabisch und Hebräisch dies hiermit definieren.

Ereignis-Behandlung

Der Vollständigkeit halber sollen hier noch die Attribute für die Ereignis-Behandlung genannt werden: `onclick`, `ondblclick`, `onmousedown`, `onmouseup`, `onmouseover`, `onmousemove`, `onmouseout`, `onkeypress`, `onkeydown`, `onkeyup`, die in nachfolgender Form notiert werden:

```
<tag-bezeichnung onmouseover="JavaScript-Anweisungen">
```

Die Verwendung von JavaScript-Skripten in einem HTML-Dokument würde den Rahmen dieses Skripts sprengen und ist daher einem gesonderten Skript vorbehalten.

3.2.3. Ausrichtattribute

- Horizontale Ausrichtung *align* (von englisch alignment):
  `<tag-bezeichnung align="Wert">`
  Mögliche Werte für das *align*-Attribut sind u.a: `center`, `left`, `right`, `justify` (Blocksatz).
- Vertikale Ausrichtung *valign* (von englisch vertical alignment):
  `<tag-bezeichnung valign="Wert">`
  Mögliche Werte für das *valign*-Attribut sind u.a: `middle`, `top`, `bottom`.

3.3. Farbdefinitionen

Farbdefinitionen gehören normalerweise nicht als Attribute in Tags; vielmehr sollen die Farben in den Formatvorlagen (Cascading Style Sheets) vorgenommen werden. Eine gewisser Kompromiss besteht im Falle des `<body>`- und der Tabellen-Tags, wo Hintergrund- bzw. Farben für Lins ausnahmsweise definiert werden dürfen.[11]

- Beispiele für Farbdefinitionen:
  - Hintergrundfarbe (Schriftfarbe): `bgcolor` (achten Sie auf die amerikanische Schreibweise).
    ```
    <tag-bezeichnung bgcolor="Farbe">
    ```

Eine Farbe wird im Regelfall durch seine Intensitätswerte der einzelnen Farbbestandteile Rot, Grün und Blau charakterisiert. Die Angabe erfolgt in der Form `#RRGGBB`, d.h., nach dem Doppelkreuz (Raute) folgen jeweils in zweiziffriger Angabe die Intensitätswerte für die drei Farbbestandteile (Wertebereich 0–255 bzw. 00–ff). Der jeweilige Intensitätswert wird in hexadezimaler[12] Weise kodiert, achten Sie darauf, dass immer zwei Ziffern geschrieben werden, führende Nullen dürfen nicht weggelassen werden. Letztendlich müssen sechs Ziffern hinter dem Doppelkreuz stehen. Insgesamt können auf diese Weise 16.777.216 verschiedene Farben definiert werden, die Ihr Computer hoffentlich darstellen kann.

Die Bestimmung der Farbwerte ist nicht ganz trivial. Um dies zu vereinfachen sind für 140 Farben Namen vordefiniert worden, die an Stelle der `#RRGGBB` angegeben werden können. Sie können im Tabellenanhang die Farbbezeichnungen erfahren.

Wenn Sie doch eine andere Farbe benötigen, so können Sie im Prinzip wie folgt vorgehen (in grafikorientierten Editoren ist meist ein Farbauswahlfenster enthalten, sodass sich die Farbauswahl einfacher gestaltet).

---

[11] Die Definition von Vordergrundfarben ist eigentlich nicht erwünscht. Dies wäre ohnehin nur für das `<font>`-Tag möglich, das aber nicht zum Einsatz kommen sollte. Eine Ausnahme stellt die fehlerhafte Farbdefinition von Horizontallinien beim Internet-Explorer dar, sie muss hier als Vordergrund-, nicht aber richtigerweise als Hintergrundfarbe in der Form `<hr color="Farbe">` angegeben werden. In beiden Fällen gilt aber, dass die Definition über Formatvorlagen erfolgen sollte!

[12] Das hexadezimale Zahlensystem ist ein Zahlensystem zu Basis 16, d.h. es gibt pro Stelle 16 Ziffern: 0–9, a–f (die Buchstabenangaben für a–f können in Groß- oder Kleinbuchstaben notiert werden). Dabei entsprechen die Buchstaben a bis f den „Ziffern“ 10 bis 15.

Wählen Sie ein Farbauswahl-Dialogfenster eines beliebigen Programms, z.B. Paint im Microsoft ® Windows ® 9x bzw. 2000. Dazu wählen Sie aus dem Menü Farben: Palette bearbeiten. Es erscheint ein Dialogfenster, in dem Sie auf den Schalter „Farben definieren“ drücken. Dann erscheint ein erweiterter Dialog:

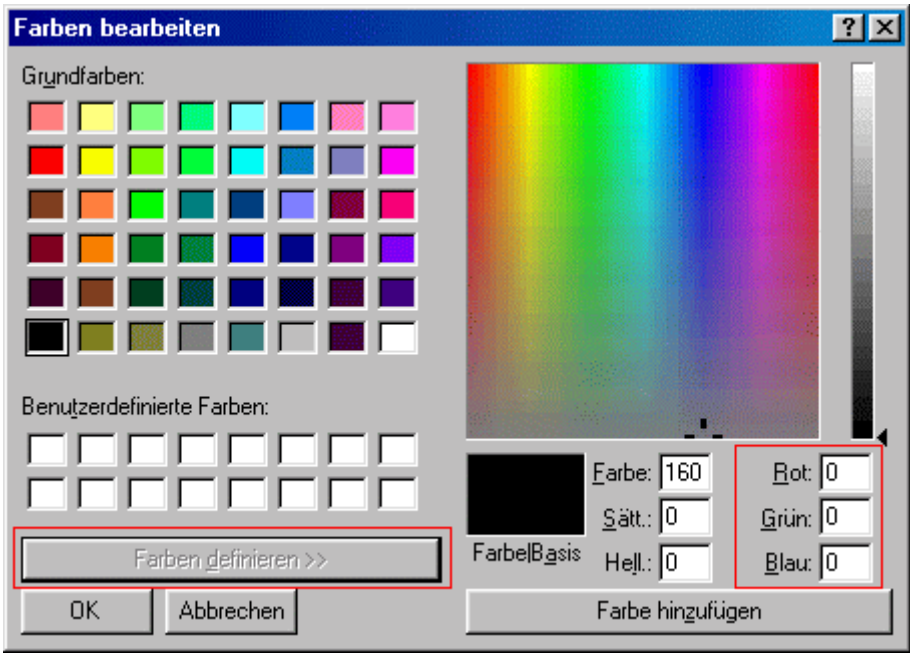

Abb. 1: Dialogfenster „Farben bearbeiten“

Mit den Schiebern suchen Sie Ihre Wunschfarbe aus. Rechts unten können Sie die Intensitätswerte für Rot, Grün und Blau im RGB-Farbsystem in dezimaler Zahlendarstellung einsehen[13].

[13] Es gibt ein zweites Farbsystem, bestehend aus Werten für Farbe, Sättigung und Helligkeit: Beide Systeme lassen sich ineinander umrechnen.

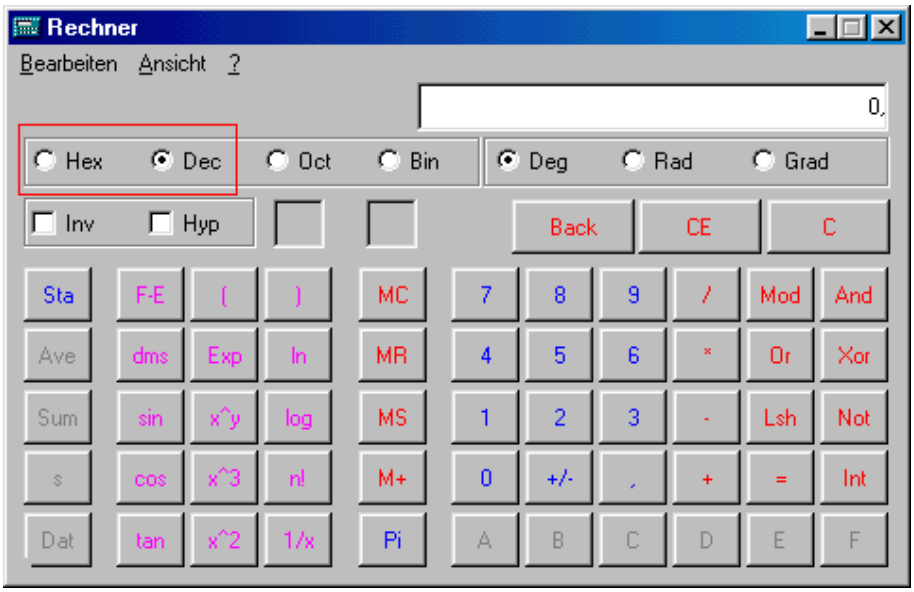

Abb. 2: Taschenrechner

Die Intensitätsangaben erfolgen in dezimaler Darstellung, Sie müssen Sie nun in hexadezimale Zahlen umrechnen. Dies können Sie z.B. mit dem Taschenrechner (Start: Zubehör: Rechner) vornehmen:

Wählen Sie Ansicht wissenschaftlich. Drücken Sie auf den Auswahlfeld „Dec“, geben Sie den dezimalen Zahlenwert ein und drücken Sie dann auf das Auswahlfeld „Hex“. Im Eingabefenster erscheint nun der Hexadezimalwert. Das Ganze macht mann für alle drei Intensitätswerte. Achten Sie bitte darauf, dass Zahlenwerten von 0 bis 15 (f) eine Null vorangestellt wird.

Beispiele:

- `<table bgcolor="#FF0000">`
- `<table bgcolor="red">`

Farbwerte können (und sollten im Regelfall) im später beschriebenen Cascading Style Sheet definiert werden.

3.4. Kommentare

Sie können an jeder beliebigen Stelle des Dokuments einen Kommentar eingeben, der nur für den Autor bestimmt ist und nur im Quelltext eingesehen werden kann. Ein Kommentar wird zwischen `<!--` und `-->` eingeschlossen. Achten Sie auf die beiden Bindestriche. Bei mehrzeiligem Kommentar schließen Sie mit `//-->` ab.

Beispiele:

- `<!-- Dies ist ein einzeiliger Kommentar. -->`
- ```
  <!-- Dies ist ein
  zweizeiliger Kommentar. //-->
  ```

Neben Kommentaren kann man hiermit zeitweiligen HTML-Code auskommentieren, d.h., der Anwender bekommt hiervon nichts mit.

Ein weiterer wichtiger Anwendungsbereich stellt das Auskommentieren von Skripten und Style-Sheet-Angaben im HTML-Kopf dar. Ältere Browser haben die Angewohnheit, diesen im Kopf stehenden Text auszugeben, obwohl dies nicht geschehen sollte: Angaben im HTML-Kopf gehören nicht auf den Bildschirm!
  ```

3.5. Buchstaben und Zeichen: Numerische Angaben und Entities

Hierunter wollen wir Platzhalter für Buchstaben und Zeichen verschiedenster Art, die der Browser sonst nicht darstellen könnte, verstehen. Die einfachste Art der Angabe – und sie ist immer einsetzbar –, ist die Angabe der zugehörigen Codenummer. Die Nummernangabe kann sowohl dezimal als auch hexadezimal erfolgen:

Beispiele:

- ` ` `<!-- Geschuetztes Leerzeichen dezimale Angabe -->`
- ` ` `<!-- Geschuetztes Leerzeichen hexadezimale Angabe -->`

Achten Sie immer auf das einleitende & (Ampersand, kaufmännisches Und) und das abschließende Semikolon.

Da man sich nicht alle Kodierungen merken kann, gibt es für ausgewählte Zeichen und Buchstaben so genannte Entities, d.h. gut einprägbare Buchstabenkürzel. Achten Sie hier unbedingt auf Groß- und Kleinschreibung. Für das o.g. Beispiel des geschützten (d.h. nicht umbrechbaren Leerraums) gilt außerdem

- ` ` `<!-- Geschuetztes Leerzeichen -->`
  `<!-- engl. non-breaking space -->`

Hierzu gibt es drei wichtige Anwendungsfälle:

- Darstellung von Zeichen, die der HTML-Syntax vorbehalten sind.
- Darstellung von Buchstaben und Zeichen im Code-Bereich von 128 bis 255 (x80 bis xFF).
- Darstellung von Buchstaben und Zeichen im UNICODE.

Darstellung von Zeichen, die der HTML-Syntax vorbehalten sind

Dies betrifft vier Zeichen:

- `&quot;` bzw. `&#34;` für das doppelte Anführungszeichen (*engl.* quotation).
- `&` bzw. `&#38;` für das kaufmännische Und (*engl.* ampersand).
- `<` bzw. `&#60;` für das kleiner-als-Zeichen („<“; *engl.* lower than).
- `>` bzw. `&#62;` für das kleiner-als-Zeichen („>“; *engl.* greater than).

Darstellung von Buchstaben und Zeichen im Code-Bereich von 128 bis 255 (x80 bis xFF)

Ursprünglich wurden alle Zeichen im Internet in einem 7-Bit-Code übertragen, d.h., damit können 128 Charakter dargestellt werden. Dies ist zwar später auf 8 Bit (d.h. 256) Charakter erweitert worden. Nichtsdestotrotz gibt es Schwierigkeiten bei der Darstellung. Wir empfehlen daher für alle Zeichen im Wertebereich von 128 bis 255 (x80 bis xFF) die Kodierung als Entity. Verzichten Sie für diesen Wertebereich auf die Verwendung von Buchstaben und Nummerncodes! Diese werden in Ländern, die nicht den westlichen Zeichensatz verstehen, unter Umständen zu fehlerhafte Zeichen: aus deutschen Umlauten werden so vielleicht arabische Schriftzeichen. Und: Verzichten Sie generell auf die Codes im Bereich 130–159.

Wichtige Beispiele:

- ` ` für nicht umbrechbarer Leerraum.
- `&sect;` für das Paragraphenzeichen („§“; *engl.* section).
- `&copy;` für das Copyright-Zeichen („©“).
- `&deg;` für das Gradzeichen zum Beispiel bei °C (*engl.* degree).
- `&plusmn;` für das Plus-Minus-Zeichen („±“).

- `&Auml;` für das große Ä.
- `&Ouml;` für das große Ö.
- `&Uuml;` für das große Ü.
- `&auml;` für das kleine ä.
- `&ouml;` für das kleine ö.
- `&uuml;` für das kleine ü.
- `&szlig;` für die ß-Ligatur.

Darstellung von Buchstaben und Zeichen im UNICODE

Es dürfte Ihnen ohne Weiteres aufgefallen sein, dass man auch mit 256 Kodiermöglichkeiten nicht alle gängigen Buchstaben schreiben kann. Dazu wurde und wird das UNICODE-Zeichensystem erarbeitet. Für die Buchstaben und Zeichen werden zwei und vier Bytes reserviert, und zumindest unter Zuhilfenahme von vier Byte sind alle Buchstaben aus heutigen, aus vergangenen und zukünftigen Tagen darstellbar. Zur Problematik UNICODE wird später nochmal eingegangen. Hier nur ein kleines Beispiel:

- `&#913;` bzw. `&Alpha;` für das große griechische Alpha.

Die UNICODE-Zeichen ab 256 dürfen in jedem Fall sowohl nummernkodiert als auch als Entity angegeben werden, da die Kodierung eindeutig ist. Es sei hier allerdings gesagt, dass natürlich nur für einen Bruchteil der Buchstaben und Zeichen Entities bereitstehen.

Im Tabellenanhang können Sie sich über weitere Belegungen informieren.

3.6. Dateinamen-Konventionen

Die Entwicklung des Internets ist eng mit dem Betriebssystem UNIX verbunden. Daher sollten Sie auf die dort gültigen Dateinamen-Konventionen achten. Die Browser unterstützen die nachfolgend beschriebenen Konventionen auch dann, wenn Sie Ihre Internet-Seiten auf einem DOS- oder Windows-Rechner erstellen.

Zum anderen kann es sinnvoll sein, eine Internetpräsenz auch offline auf einer CD abrufen zu können. Hier sollten die Dateinamen-Konventionen, wie sie in der ISO-9660 spezifiziert sind, ebenfalls eingehalten werden.

Daher sollten Sie Folgendes beachten:

- Benutzen Sie im Dateinamen nur Buchstaben des englischen Alphabets (a-z), die Ziffern 0-9, den Unterstrich "_" und den Punkt. Benutzen Sie in keinem Fall Umlaute und das ß!
- Die unterschiedlichen Betriebssysteme behandeln Groß- und Kleinbuchstaben in Dateinamen unterschiedlich (als identische oder nicht identische Buchstaben). Es ist sinnvoll, für Dateinamen ausschließlich Kleinbuchstaben zu verwenden, ein entsprechender Umwandlungsmodus wird von wichtigen FTP-Programmen unterstützt. (Auch wenn man es nicht sieht, pflegt Windows den ersten Buchstaben von neu angelegten Dateien groß zu schreiben.)
- Bei Dateinamen sollte eine Länge von 64 Zeichen, besser noch eine Länge von 31 Zeichen (ISO-9660-(Level 2)-Format) nicht überschritten werden!

UNIX besitzt ein sehr elegantes System von relativen Verzeichnisangaben:

- `./dir` bzw. `./datei` bedeutet ein Bezug vom momentanen Verzeichnis oder von der aktuellen Datei aus. Nehmen wir an, wir oder unsere Datei befänden uns im Verzeichnis `/dir1/dir2`, so würde ein Relativbezug `./dir3/datei.exe` bedeuten, dass die Datei dann `/dir1/dir2/dir3/datei.exe` heißt.
- `../` bedeutet eine Dateiebene zurück. Wenn wir uns im Verzeichnis `/dir1/dir2/dir3` befinden, bedeutet `../../dir4/datei`, dass die Datei `/dir1/dir4/datei` heißt.
- `/` bedeutet Wurzelverzeichnis, d.h., das für die Anwendung oder den Nutzer festgelegte Startverzeichnis. Sie sollten diese Angabe nicht benutzen, weil dies eine absolute, und keine relative Verzeichnisangabe einleitet.
- Eine Datei ohne Verzeichnisangabe wird immer nur im aktuellen Verzeichnis gesucht!

Den Vorteil relativer Adressierung lernt man spätestens dann schätzen, wenn Sie ganze Verzeichnisbäume in andere Unterverzeichnisse verschieben: Alle Bezüge klappen noch, soweit Sie sich auf die verschobene Struktur beziehen.

4. Beschreibung einzelner Tags

4.1. Aufbau einer HTML-Seite

Wie nachfolgendes kleines Quellkodebeispiel verdeutlicht, besteht eine HTML-Datei aus:

- der Dokumenttypdefinition (Beschreibung in separatem Kapitel),
- dem mit `<html>` ... `</html>` vorgegebenen Rahmen,
- und Kopf- und Rumpfteil (`<head>`...`</head>` bzw. `<body>`...`</body>`).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
   "http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
<head>
   <title>Der erste Versuch</title>

   <meta http-equiv="Content-Type"
      content="text/html; charset=iso-8859-1">

   <meta name="description"
      content="&Uuml;bung HTML-Grundger&uuml;st: Wirklich
      nur der erste Versuch.">
   <meta name="keywords" content="HTML, Grundger&uuml;st">
   <meta name="author" content="Roland Unger">
</head>

<body>

<p>Noch nicht viel. Aber das wird schon.</p>

</body>
</html>
```

Für ein XHTML-Dokument sähe das Grundgerüst z.B. wie folgt aus:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 4.0 Transitional//EN"
   "DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de" >
<head>
   <title>Der erste Versuch</title>

   <meta http-equiv="Content-Type"
      content="text/html; charset=iso-8859-1" />

   <meta name="description"
      content="&Uuml;bung HTML-Grundger&uuml;st: Wirklich
```

```
      nur der erste Versuch." />
   <meta name="keywords" content="HTML, Grundger&uuml;st" />
   <meta name="author" content="Roland Unger" />
</head>

<body>

<p>Noch nicht viel. Aber das wird schon.</p>

</body>
</html>
```

Der Kopfteil enthält dokumentspezifische Definitionen und ist deshalb für den Betrachter nicht sichtbar. Der Rumpfteil enthält all die Informationen, die der Betrachter zu sehen bekommen soll.

Streng genommen gehören natürlich das `<title>`- und das `<meta>`-Tag nicht zum Grundgerüst, Sie erkennen somit aber, welche Angaben im Kopf notiert werden.

Der Titel wird in der Regel in der Programmzeile des Browsers angezeigt. Bitte wählen Sie einen möglichst aussagekräftigen Titel, vermeiden Sie Titel wie „Willkommen auf meiner Homepage“ oder “Unknown Title”.

Anhand der ersten Tags erkennen Sie deutlich die Verwendung von einleitenden und schließenden Tags. Das `<body>`-Tag markiert bereits den ersten nutzbaren Text-Container, d.h., man könnte theoretisch auf Absätze verzichten, was aber nur bei kleinen Texten einen Sinn macht.

`<html> ... </html>`

- Kennzeichnet Anfang und Ende eines HTML-Dokuments.
- `<html>` besitzt außer der Angabe der Dokumentsprache keine Attribute.
- Unter XHTML 1.0 besitzt `<html>` folgende Attribute:
  - `xmlns`: Gibt ein Gerät an, dass zur Deklaration von Namenräumen (name space) dient. Auch wenn die Angabe bisher mit nichts verbunden ist, ist die Angabe *obligatorisch*. Es ist nicht sinnvoll, gegenwärtig von der o.g. Angabe abzuweichen.
  - `xml:lang`: Gibt die Dokumentsprache an.
  - `lang`: Gibt die Dokumentsprache an (Abwärtskompatibilität für nicht-XML-fähige Browser).

```
<head> ... </head>
```

- Kennzeichnet Anfang und Ende des Kopfes.
- `<head>` besitzt keine Attribute.

```
<body> ... </body>
```

- Kennzeichnet Anfang und Ende eines Textkörpers.
- `<body>` besitzt Attribute, keins der Attribute ist aber zwingend notwendig.
- Mögliche Attribute:
  - `text="farbe"` Farbe des Textes
  - `bgcolor="farbe"` Farbe des Hintergrundes. Der Hintergrund könnte allerdings durch ein Hintergrundbild überdeckt werden.
  - `background="bild"` URL einer Bilddatei (*.gif oder *.jpeg) für den Hintergrund.
  - `link="farbe"` Farbe eines noch nicht verfolgten und nicht aktivierten Links.
  - `vlink="farbe"` Farbe eines bereits verfolgten, aber nicht aktivierten Links.
  - `alink="farbe"` Farbe eines aktivierten Links.
- Diese Definitionen sollten aber mit Cascading Style Sheets vorgenommen werden.

Es sei bereits hier darauf hinzuweisen, im Interesse einer einfachen Änderbarkeit des Formats auf die hier genannten Attribute zu verzichten und die Definitionen in einer Formatvorlage vorzunehmen.

4.2. Dokumenttypdefinitionen

Auch wenn heutzutage noch kein Browser eine Dokumenttypdefinition auswertet, so wird sie unter XML zwingend notwendig. Außerdem ist es denkbar, dass zukünftige Browser eine Analyse sehr wohl vornehmen und die Darstellung entsprechend danach ausrichten.

HTML-Sprachstandards 2.0 und 3.2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 2.0//EN">
```
Damit spezifizieren Sie den HTML-Sprachstandard 2.0.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
```
Damit spezifizieren Sie den HTML-Sprachstandard 3.2 als den Standard Ihrer HTML-Datei.

Denken Sie daran, dass der Standard für die Version 3 die Nummer 3.2 besitzt.

HTML-Sprachstandard 4.0

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
```
Damit spezifizieren Sie den HTML-Sprachstandard 4.0 als den Standard Ihrer HTML-Datei.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```
Damit spezifizieren Sie den HTML-Sprachstandard 4.0 und geben an, dass Sie außerdem Style-Sheets und/oder Scriptsprachen in der Datei einsetzen.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
```
Damit spezifizieren Sie den HTML-Sprachstandard 4.0 und geben an, dass Sie in der HTML-Datei ein Frame-Set definieren.

XHTML-Sprachstandard 1.0

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "DTD/xhtml1-strict.dtd">
```
Damit spezifizieren Sie den XHTML-Sprachstandard 1.0 als den Standard Ihrer XHTML-Datei. Diese Definition wird benutzt, wenn Sie eine tatsächlich sauber strukturierte Auszeichnungssprache benutzen, ohne dass Tags benutzt werden, die das Layout betreffen. Layout-Definitionen dürfen nur über Cascading Style Sheets vorgenommen werden.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "DTD/xhtml1-transitional.dtd">
```
Damit spezifizieren Sie den XHTML-Sprachstandard 1.0 und geben an, dass Sie außerdem Style-Sheets und/oder Scriptsprachen in der Datei einsetzen. Weiterhin dürfen Sie Layout-Angaben wie die `<body>`-Attribute `bgcolor`, `text` und `link` verwenden.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
   "DTD/xhtml1-frameset.dtd">
```
Damit spezifizieren Sie den XHTML-Sprachstandard 1.0 und geben an, dass Sie in der XHTML-Datei ein Frame-Set definieren.

4.3. Gliederung und Textblöcke

4.3.1. Einführung

Bei der Anwendung von Formatierungen müssen wir unterscheiden, ob sie zeichen- oder absatzgebunden sind.

Typische Beispiele für zeichengebundene Formatierung sind Attribute wie fett oder kursiv. Der Urahn aller derartigen Tags ist das `<span>`-Tag.

Typische Beispiele für absatzgebundene Formatierung sind Überschriften, Horizontallinien, und die verschiedenen Absatz-Tags (z.B. `<p>`, `<pre>`). Urahn aller derartigen Tags ist das `<div>`-Tag.

Wörter können durch Leerräume oder Zeilenumbruch getrennt werden. Auch wenn mehrere Leerräume in der HTML-Datei hintereinander stehen, so wird nur ein Leerraum dargestellt. Achten Sie, insbesondere im Falle von unterstrichenen Texten und rechtsbündigen Texten, darauf, dass das schließende Tag unmittelbar hinter dem letzten Wort steht: fügen Sie hier keinen weiteren Leerraum oder Zeilenumbruch ein: sie werden dargestellt.

Abschreckende Beispiele:

Dies ist ein Link. Nun der nächste Satz. Rechtsbündiger Text

Die unterste Zeile des rechtsbündigen Textes ist nicht mehr rechtsbündig: es ist noch ein Leerraum dazwischen!

4.3.2. Überschriften

Es gibt die sechs Tags `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` und `<h6>`, die ein hierarchisches Überschriftensystem darstellen. Dabei sind Überschriften mit dem `<h1>`-Tag die bedeutendsten Hauptüberschriften, während `<h6>` die Unterüberschrift der niedrigsten Hierarchieebene darstellen. Dabei müssen Sie nicht immer mit `<h1>` beginnen, wenn Ihnen der gewählte Schrifttyp zu groß erscheint, auch kann man eine Hierarchieebene auslassen. Achten Sie bitte darauf, dass Sie `<h1>` mit `</h1>`, `<h2>` mit `</h2>` usw. abschließen.

Allerdings gibt es im HTML kein (einfaches) Mittel, aus diesen Überschriften ein Inhaltsverzeichnis zu erstellen.

```
<h1> ... </h1>
```

- Kennzeichnet Anfang und Ende einer Überschrift der Hierarchieebene 1.
- Optionale Attribute für `<h1>` sind die bereits beschriebenen Universalattribute, *valign* ist nicht einsetzbar.
- Beispiel: `<h1 align="center">&Uuml;berschrift</h1>`

4.3.3. Horizontallinie

Ohne Attribut wird eine horizontale Linie gezeichnet, standardmäßig zentriert. Über das `width`-Attribut lässt sich die Länge absolut und prozentual festlegen.

```
<hr>
```

- Kein schließendes Tag. In XHTML als `<hr ... />` anzugeben.
- Neben Universalattributen kann das Attribut `width` optional eingesetzt werden. Beispiele:
  - `<hr width="150">` (absolute Breite 150 Pixel)
  - `<hr width="70%">` (prozentuale Breite 70%)

4.3.4. Absatzgestaltung

`<p> ... </p>`

- Kennzeichnet Anfang und Ende eines Absatzes.
- Optional kann das Attribut `align` für die Textausrichtung benutzt werden (mögliche Werte `left`, `right`, `center`, `justify` (Blocksatz).
- Beispiel: `<p align="justify">Blocksatz.</p>`

`<blockquote> ... </blockquote>`

- Kennzeichnet Anfang und Ende eines Absatzes eines Zitats. Gegenüber dem üblichen Absatz `<p>` ... `</p>` entsteht ein Absatz, der auf der rechten und linken Seite eingerückt ist.
- Optional kann das Attribut `align` für die Textausrichtung benutzt werden (mögliche Werte `left`, `right`, `center`, `justify` (Blocksatz).
- Beispiel: `<blockquote align="justify">Zitat.</blockquote>`

`<pre> ... </pre>`

- Kennzeichnet Anfang und Ende eines Absatzes für so genannten vorformatierten Text. Darunter versteht man Quell- oder vergleichbare Texte, die mit Buchstaben konstanter Breite dargestellt werden sollen. Weiterhin erfolgt ein Zeilenumbruch ohne entsprechende Tags, ein bestehender Zeilenumbruch wird ausgeführt!
- Das optionale Attribut `width="Breite_in_Charaktern"` bewirkt einen Zeilenumbruch nach der angegebenen Anzahl von Buchstaben oder Zeichen.
  - Beispiel: `<pre width="5">abcdefghijkl</pre>`
  - Aussehen:
    ```
    abcde
    fghij
    kl
    ```

`<div> ... </div>`

- Ein Absatz-Tag, das in seinem Aussehen dem `<p> ... </p>`-Tag ähnelt. Es verfügt aber über eine erweiterte Funktionalität gegenüber dem üblichen Absatz-Tag:
  - Es ist eine Art Über-Container, der den eingebetteten Tags das Ausrichtattribut `align` aufzwingen kann. Das vermeidet unnötige Schreibarbeit. Beispiel:

    - ```

         <h1>Titel</h1>
         1. Absatz
         2. Absatz

      ```
    - Alle Blöcke werden zentriert ausgerichtet.
- Es ist im Rahmen des dynamischen HTML (DHTML) ein auf dem Bildschirm frei positionierbares und modifizierbares Blockelement.
- Es kann als Ausgangs-Element für verschiedene **Block-Elemente** benutzt werden, die über Cascading Style Sheets definiert werden können. Im gewissen Maße ist das `<div>`-Tag der Urahn der `<p>`-, `<blockquote>`- und `<pre>`-Tags. Das `<span>`-Tag wird eine solche Funktion für Elemente innerhalb eines Blocks einnehmen und für Textauszeichnungen zuständig sein.

4.3.5. Zeilenumbruch

```
<br>
```

- Bewirkt einen festen Zeilenumbruch.
- Im XHTML als `<br />` anzugeben.
- Einige Browser führen das `<br>`-Tag nur dann aus, wenn noch weiterer Text folgt. Für den Fall einer Lehrzeile sollte man dem Tag einen nichtumbrechbaren Leerraum folgen lassen: `<br> `

```
 
```

- Der Charakter für einen nichtumbrechbaren Leerraum.
- Beispiel: `3,5 m` für 3,5 m. Es wird hiermit verhindert, dass die Maßeinheit evtl. umgebrochen wird
- Dieser Leerraum wird bei der Ausrichtung `align="justify"` nicht mit in die Länge gezogen.

Siehe auch die Anmerkungen im Anhang zum Thema: „Ersatz von Tags zum Zeilen- bzw. Wortumbruch“ auf S. 103.
```

4.4. Textauszeichnungen

Im HTML sind sowohl Tags für physische als auch logische Textauszeichnung vorhanden:

HTML-Tags für physische Textauszeichnung

`<b> ... </b>`	bewirkt **fett formatierten Text**.
`<i> ... </i>`	bewirkt *kursiv formatierten Text*.
`<tt> ... </tt>`	bewirkt `dicktengleich formatierten Text (Teletyper = Fernschreiber)`.
`<u> ... </u>`	bewirkt unterstrichenen Text.
`<strike> ... </strike>`, `<s> ... </s>`	bewirkt ~~durchgestrichenen Text~~.
`<big> ... </big>`	bewirkt groß formatierten Text.
`<small> ... </small>`	bewirkt kleiner formatierten Text.
`<sup> ... </sup>`	bewirkt $^{\text{hochgestellten}}$ Text.
`<sub> ... </sub>`	bewirkt $_{\text{tiefgestellten}}$ Text.

Die Tags `<u> ... </u>`, `<strike> ... </strike>` und `<s> ... </s>` stehen auf der Abschussliste des W3-Konsortiums. Hier hilft die Verwendung von Cascading Style Sheets.

HTML-Tags für logische Textauszeichnung

Der Unterschied zur physischen Textauszeichnung besteht darin, dass die Formatierung erst durch den Browser oder durch eine Formatvorlage festgelegt wird.

Allerdings wird in Zukunft der Unterschied zwischen logischer und physischer Textauszeichnung verschwinden, da jedem Tag eine Formatvorlagendefinition vorgegebenen werden kann. Das `<b> ... </b>` müßte sogar nicht mehr eine Fett-Formatierung vornehmen.

`<em> ... </em>`	formatiert einen *Text mit der Bedeutung „betont (emphatisch)“*.
`<strong> ... </strong>`	formatiert einen **Text mit der Bedeutung „stark betont“**.
`<code> ... </code>`	formatiert einen `Text mit der Bedeutung „dies ist Quellcode“`.
`<samp> ... </samp>`	formatiert einen `Text mit der Bedeutung „dies ist ein Beispiel“`.

`<kbd> ... </kbd>`	formatiert einen `Text mit der Bedeutung „dies ist eine Tastatureingabe“`.
`<var> ... </var>`	formatiert einen *Text mit der Bedeutung „dies ist eine Variable“*.
`<cite> ... </cite>`	formatiert einen *Text mit der Bedeutung „dies ist ein Zitat aus einer anderen Quelle“*.

Folgende logischen HTML-Tags stehen seit HTML 4.0 zur Verfügung

`<abbr> ... </abbr>`	formatiert einen Text mit der Bedeutung „dies ist eine Abkürzung“. (z.B. „Abk.“) Der Text erhält eine punktierte Unterstreichung, die Erläuterung der Abkürzung wird einem kleinen Tool-Tip-Fenster beim Überfahren mit der Maus angezeigt.
`<acronym>....</acronym>`	formatiert einen Text mit der Bedeutung „dies ist ein Akronym“. (z.B. „VIP“) Der Text erhält eine punktierte Unterstreichung, die Erläuterung des Akronyms wird einem kleinen Tool-Tip-Fenster beim Überfahren mit der Maus angezeigt.
`<dfn>....</dfn>`	*formatiert einen Text mit der Bedeutung „dies ist eine Definition“*.
`<q cite="URI">....</q>`	formatiert einen Text mit der Bedeutung "dies ist ein Zitat mit Quellenangabe". Das `<q>`-Tag unterscheidet sich vom `<blockquote>`-Tag dahin gehend, dass es innerhalb eines Absatzes angewendet wird. Über die mögliche Ausgabeform des Zitats gibt es noch keine Festlegungen.

Tags mit Sonderfunktionen

`<del>....</del>`	Änderungsmarkierung für ~~für~~ zu löschenden Text.
`<ins>....</ins>`	Änderungsmarkierung für einzufügenden Text.
`<bdo dir="rtl">....</bdo>`	Markierung für Textrichtungsumkehr: rhekmusgnuthcirtxeT.

Vorgriff auf Cascading Style Sheets

`<span> ... </span>`

- Ohne Attribute bewirkt dieses Tag überhaupt keine Änderung der Textauszeichnung. Dieses Tag ist zusammen mit den Cascading Style Sheets eingeführt worden.
- Mindestens eins der nachfolgenden Attribute muss spezifiziert werden:

  - `style="Stilanweisungen"`. Stilanweisungen können beliebige, im Kapitel Cascading Style Sheets zu besprechende Stildefinitionen sein.
  - `class="Klassenname"`. Bewirkt die Übernahme von Textauszeichnungen der der Klasse „Klassenname“ zugeordneten Stildefinitionen.
- Ähnlich dem `<div> ... </div>` besitzt das `<span> ... </span>`-Tag eine zentrale Bedeutung für die Auszeichnung von Elementen innerhalb von Blöcken, denen man über kein anderes Tag eine Stilangabe zuordnen kann. Im gewissen Maße ist das `<span>`-Tag der Urahn der oben genannten physischen und logischen Textauszeichnungen.

Das in der Vergangenheit häufig eingesetzte Tag `<font>` zur Definitionen von Schriftarten, -größen und -farben ist nicht standardisiert und sollte daher durch das `<span>`-Tag oder durch Stildefinitionen ersetzt werde.

4.5. Listen

Nachfolgende Listenformen stehen Ihnen zur Verfügung:

4.5.1. Geordnete Listen (nummerierte Listen) – Ordered Lists

```
<ol type="type" start="start_wert">
   <li value="aktueller_wert">Listenelement</li>
   <!-- weitere Listenelemente -->
</ol>
```

Die Liste wird durch das `<ol>`-Tag umschlossen. Das `<li>`-Tag kennzeichnet den Listeneintrag.

`<ol> ... </ol>`

- Optionale Attribute:
  - `type="a|A|i|I"`: Typ des Nummerierungssystems:
    - Ohne Typangabe: Zehnersystem mit arabischen Ziffern.
    - `type="a"`: Mit Kleinbuchstaben. Das Zahlensystem ist ein System zur Basis 26: 2 wird zu b, 26 zu z, 27 zu aa, 54 zu ba usw.
    - `type="A"`: Mit Großbuchstaben. Das Zahlensystem ist ein System zur Basis 26.
    - `type="i"`: Mit kleinen römischen Ziffern.
    - `type="I"`: Mit großen römischen Ziffern.
  - `start="start_wert"`: Zahlenwert des ersten Listeneintrags.

`<li> ... </li>`

- list item.
- Optionales Attribut:
  - `value="wert"`: Hiermit geben Sie einen Wert an, mit dem die Nummerierung fortgesetzt werden soll.

Achten Sie darauf, dass bei großen Zahlen sowie bei Verwendung römischer Ziffern der für die Zahl reservierte Platz möglicherweise nicht ausreicht und das Layout darunter leidet.

4.5.2. Nichtgeordnete Listen (Aufzählungslisten, Bullet-Listen) – Unordered Lists

```
<ul type="type">
   <li>Listenelement</li>
   <!-- weitere Listenelemente -->
</ul>
```

Die Liste wird durch das `<ul>`-Tag umschlossen. Das `<li>`-Tag kennzeichnet den Listeneintrag. Die Liste erhält als Kennzeichnung der Listeneinträge graphische Symbole wie Kreise, Scheiben und Quadrate.

`<ul> ... </ul>`

- Optionale Attribute:
  - `type="circle|disc|square"`: Type des graphischen Symbols:
    - Ohne Typangabe: Standardform. In der ersten h´Hierarchie zumeist ein scheibenförmiges Symbol
    - `type="circle"`: Als Symbol wird ein Kreis dargestellt.
    - `type="disc"`: Als Symbol wird eine Scheibe dargestellt.
    - `type="square"`: Als Symbol wird ein Quadrat dargestellt.

`<li> ... </li>`

- Es gibt (natürlich) kein `<li>`-Tag-spezifisches Attribut.

4.5.3. Definitionslisten

Definitionslisten sind eigentlich für Indexverzeichnisse Gedacht: Eine Struktur, die aus Hauptstichwörtern und Unterstichwörtern besteht. Somit gibt es zwei verschiedene Listeneintragstypen. Die Liste wird durch das `<dl>`-Tag umschlossen. `<dt>` und `<dd>` kennzeichnen die Listeneinträge.

```
<dl>
   <dt>definition term, Hauptstichwort</dt>
   <dd>definition (term) definition, Unterstichwort</dd>
   <!-- weitere Listeneinträge -->
</dl>
```

```
<dl> ... </dl>
```

- Es gibt außer den Universalattributen keine weiteren Attribute.

```
<dt> ... </dt>
```

- definition term. Dies ist das Hauptstichwort. Es wird nicht eingerückt.
- Es gibt außer den Universalattributen keine weiteren Attribute.

```
<dd> ... </dd>
```

- definition (term) definition. Dies ist das Unterstichwort. Es wird eingerückt.
- Es gibt außer den Universalattributen keine weiteren Attribute.

4.5.4. Weitere Listen

Nachfolgende Listen sind wenig gebräuchlich und werden daher nur kurz vorgestellt:

```
<menu>
   <li>Listenelement</li>
   <!-- weitere Listenelemente -->
</menu>

<dir>
   <li>Listenelement</li>
   <!-- weitere Listenelemente -->
</dir>
```

Sie werden von den heutigen Browsern entweder nicht mehr oder nicht anders dargestellt als gewöhnliche Aufzählungslisten. Dennoch könnten diese Befehle für künftige Anwendungsbereiche von (X)HTML in Zusammenhang mit Cascading Style Sheets wieder an Bedeutung gewinnen.

4.5.5. Verschachtelung von Listen

Oben genannte Listen können beliebig geschachtelt werden. Es gibt aber so manch einen Browser, der ab der 3. Hierarchieebene mit der Zählung in Schwierigkeiten kommt. Die Schachtelungsmöglichkeit bezieht sich nicht nur auf den jeweils gleichen Listentyp, die Listentypen können ebenfalls wahlfrei gewechselt werden:

```
<ol>
   <li>Erster Eintrag</li>
   <li>Zweiter Eintrag
      <ol type="i">
         <li>Erster Eintrag</li>
         <li>Zweiter Eintrag
            <ul>
              <li>Erster Eintrag</li>
              <li>Zweiter Eintrag</li>
              <li>Dritter Eintrag</li>
            </ul>
         </li>
         <li>Zweiter Eintrag</li>
      </ol>
   </li>
   <li>Dritter Eintrag</li>
</ol>
```

Aussehen:

1. Erster Eintrag
2. Zweiter Eintrag
   i Erster Eintrag
   ii Zweiter Eintrag
      - Erster Eintrag
      - Zweiter Eintrag
      - Dritter Eintrag
   iii Dritter Eintrag
3. Dritter Eintrag

4.6. Verweise (Hyperlinks)

Die Verweise machen das Internet eigentlich erst zu dem, was es ist: ein weltweites Hypertextsystem. Dabei kann aber nicht nur zu einer beliebigen Quelle gesprungen werden, sondern auch ein bestimmtes Ziel innerhalb der HTML-Datei gleich erreicht werden.

Die Syntax des Verweises lautet:

```
<a href="url" [target="ziel"] [accesskey="charakter"]>Verweistext</a>
```

- Obligatorisches Attribut:
  - `href="url"`: Zielangabe entsprechend nachfolgender Tabelle.
- Optionale Attribute:
  - `target="ziel"`: Wird im Zusammenhang mit Frames benötigt und bewirkt die Ausgabe der Datei in einem neuen Browser- oder Frame-Fenster. Beschreibung im Kapitel Frames. Ziel darf nur aus Buchstaben, Unterstrich und Zahlen bestehen und muss mit einem Buchstaben oder dem Unterstrich beginnen.
  - `accesskey="charakter"`: Zeichen auf der Tastatur, das der Anwender drücken kann, um den Verweis direkt anzuspringen (access key = Zugriffstaste). Benutzen Sie nur Tasten, die auf jeder Tastatur zu finden sind. Diese Möglichkeit ist bisher jedoch kaum bei einem Browser implementiert.
  - `title="Kurzbeschreibung"`: Hier kann eine kurze Erläuterung wie „Link führt zu ...“ gegeben werden. Viele Browser geben diese Information beim Überfahren des Links in einem Tool-Tip-Fenster aus.
  - `tabindex="indexnummer"`: Im Falle von Links und Formularelementen können Sie Attribut zur Festlegung einer Indexnummer benutzen, die die Reihenfolge des Anspringens unter Nutzung der Tabulator-Taste festlegt. Der Link mit der niedrigsten Nummer wird zuerst angesprungen, der mit der höchsten zuletzt.

Art	Syntax
Datei auf lokalem Rechner	**file:**//[/laufwerk|]verzeichnis/[datei.html[#Ankername]]
WWW-Adresse	**http:**//rechner[:port]/[verzeichnis/[datei.html[#Ankername]]
WWW-Adresse, gesicherte (verschlüsselte Datenübertragung)	**https:**//rechner[:port]/[verzeichnis/[datei.html[#Ankername]]
Gopher-Adresse	**gopher:**//rechner/[verzeichnis/[datei]]

Art	Syntax
FTP-Adresse	**ftp:**//rechner/[verzeichnis/[datei]]
Telnet-Adresse	**telnet:**//rechner
Email	**mailto:**emailadresse[?Zusatz1[&Zusatz2[&Zusatz3]]]
News-Adresse	**news:**rechner
Datei im selben Verzeichnis (relative Adressierung)	datei.html[#Ankername]
Datei in anderem Verzeichnis (relative Adressierung)	./verzeichnisstruktur/datei.html[#Ankername] ../verzeichnisstruktur/datei.html[#Ankername]
Anker in der gleichen Datei	#Ankername

In jeder HTML-Datei können in der folgenden Weise Anker vereinbart werden. Es ist darauf zu achten, dass die Namen eindeutig vergeben werden:

HTML: `<a name="bezeichnung"></a>`
XHTML: `<a id="bezeichnung"></a>`

Wenn ein anderer Dienst als der HTTP-Dienst aufgerufen wird, so wird durch den Browser automatisch das passende Client-Programm aufgerufen. Allerdings sind heute alle modernen Browser in der Lage, alle Internet-Protokolle selbst verarbeiten zu können.

Email-Verweise

Wie in obiger Tabelle erwähnt, lässt sich mit einem Verweis ebenfalls der Email-Client aufrufen und mit Vorgabewerten versehen:

```
<a href="mailto:email@domain[?Zusatz1[&Zusatz2[&Zusatz3]]]">
```

Die Zusätze sind optional. Dem erste Zusatz wird ein Fragezeichen vorangestellt, jeder weitere Zusatz wird über ein & vermittelt. Mögliche Zusätze:

- `cc=name2@domain`: Weitere Emailadresse (Empfänger einer Kopie [carbon copy] der Email).

- `bcc=name3@domain`: Weitere Emailadresse (Empfänger einer zweiten Kopie [blind carbon copy] der Email).
- `subject=betreff`: Betreff der Email.
- `body=inhalt_der_mail`: Text für den Textkörper der Email.

Hierzu einige Anmerkungen:
- Der Link ist nur mit korrekt eingerichtetem Client nutzbar.
- Nicht jeder Email-Client kann alle Zusätze verarbeiten.
- Die zu den Zusätzen zugehörigen Werte werden ohne Anführungszeichen an das Gleichheitszeichen geschrieben.
- Die Texte des `subject`- und des `body`-Zusatzes dürfen keine Leerräume und Sonderzeichen enthalten. Wenn derartige Zeichen benötigt werden, so müssen sie hexadezimal kodiert hinter einem Prozentzeichen angegeben werden: für den Leerraum `%20`, für einen Zeilenumbruch `%0D%0A`. Letztendlich sieht es so aus: Sehr%20geehrte%20Frau%20Name.

In einem Verweis-Tag brauchen aber nicht nur HTML-Dateien zu stehen. Wenn der Browser oder eins der in ihm eingefügten Plugins die Datei selbst anzeigen/darstellen/aufführen können, so tun sie es. Ansonst bieten die neueren Browser die Möglichkeit der Übergabe an ein Programm, das diese Datei verarbeiten kann, an oder das Abspeichern auf Festplatte.

Somit ist es heute möglich, Video- und Audio-Dateien in ein Internet-Projekt mit zu integrieren.

Die Weitergabe von Dokumenten an die zugehörigen Programme (Textverarbeitungssysteme, Datenbanken u.a.) birgt aber auch die Gefahr einer möglichen Schädigung durch Virenprogramme jeglicher Art.

4.7. Eingebettete Medienobjekte

4.7.1. Grafik und Image-Map

Seit den ersten HTML-Versionen wird die Einbettung von Grafiken unterstützt. Da die Übertragungsbandbreite im Internet aber relativ gering ist, kommen hier nur Dateien mit komprimierten Bilddaten in Frage:

Dateityp	Beschreibung
*.gif	Graphics Interchange Format. Zur Komprimierung wird der lizenzpflichtige Lempel-Zev-Welch-(LZW)-Mechanismus[14] verwendet.
*.png	Portable Network Graphics.
*.jpg, *.jpeg	Grafikformat der Joint Photographic Experts Group. Komprimierungsstandard für unbewegte Bilder.

Bei den ersten beiden Verfahren erfolgt eine Komprimierung ohne Datenverlust, d.h., die entsprechenden Dateien können wieder in ihren ursprünglichen Zustand zurückgerechnet werden. Diese Formate eignen sich besonders für Bitmaps mit wenigen und großflächig verwendeten Farben, z.B. 16-farbige Piktogramme. Das *.png-Format wird aber erst von wenigen Browsern unterstützt. Beide Formate unterstützen Transparenz von Bildteilen und animierte Bilder (Folge von Bildern, die nach einer vorgebbaren Zeit ausgetauscht werden).

Bei dem letzten Verfahren wird versucht, die Detail- und Farbinformation in einem vorgebbaren Maße zu reduzieren, um den benötigten Speicherplatz zu reduzieren. Dieses verfahren bietet sich besonders bei Fotografien an.

Im Zweifelsfalle sollte man das Ursprungsbild in beiden Formate umwandeln und sich für das platzsparendste oder qualitativ beste Resultat entscheiden.

Der Aufruf eines solchen Bildes in einer HTML-Datei ist einfach:

[14] Es entstehen keine Lizenzkosten bei nichtkommerziellem Einsatz des Dateiformats. Bei gewerblicher Nutzung muss eine Lizenzgebühr entrichtet: Bei einem Grafikprogramm, das dieses Format unterstützt, werden mit dem Kaufpreis auch die Lizenzgebühren abgegolten.

```
<img src="url" alt="textangabe" width="breite"
   height="hoehe" border="randbreite" lowsrc="url" />
```

- Obligatorische Attribute:
  - `src="url"`: Quelle der Grafik-Datei.
  - `width="breite"`: Breite der darzustellenden Grafik-Datei in Pixeln.
  - `height="hoehe"`: Höhe der darzustellenden Grafik-Datei in Pixeln.
    - Die angegebenen Werte für Breite und Höhe müssen nicht mit den wirklichen Werten der Grafik-Datei übereinstimmen. Die vorhandene Grafik wird bei Notwendigkeit auf die neuen Maße vergrößert bzw. verkleinert.
    - Verzichten Sie auf Verkleinerungen und bieten Sie von vornherein kleinere speicherplatzsparendere Grafik-Dateien an.
  - `alt="textangabe"`: Hier geben Sie eine Kurzbeschreibung der Grafik an. Auch wenn der Betrachter Ihrer Seiten keine Bilder anzeigen lässt oder einen textbasierenden Browser nutzt, hat er doch somit einen Hinweis auf den Inhalt Ihrer Grafik.
- Optionale Attribute:
  - `border="randbreite"`: Breite in Pixeln eines Randes um das Bild. Standardgemäß ist der Rand bei Bildern, die nicht als Link dienen, Null. `border="1"` bewirkt das Umranden des Bildes mit einer feinen Linie.
  - `lowsrc="url"`: Hier kann man alternativ eine Grafik-Datei gleichen Inhalts, aber geringerer Auflösung angeben, die dann zuerst angezeigt wird.
  - `longdesc="url"`: Hier können Sie einen Link zu einer Langbeschreibung für das Bild angeben.
- Der Schrägstrich am Ende ist für den Einsatz unter XHTML notwendig, da das `<img>`-Tag kein schließendes Tag besitzt.

Wenn keine Bilder vom Browser ausgegeben werden (z.B. bei textbasierende Browsern), wird anstelle des Bildes der Text des `alt`-Attributes ausgegeben. Wenn das Bild Teil einer Textzeile ist, versuchen Sie einen Text oder ein Symbol zu wählen, die einen möglichst natürlichen Lesefluss ermöglichen: Verwenden Sie keine Umklammerungen wie [Bild]; Pfeile oder Stichpunkte lassen sich z.B. mit „<“, „o“, „+“, „–“ etc. umschreiben.

Die Beschriftung von Grafiken ist eigentlich nicht vorgesehen. Strebt man eine ganz bestimmte Anordnung von Grafik und Beschriftung an, so sollte man sich Tabellen bedienen.

Eine Grafik kann auch als Link benutzt werden: Dazu muss man sie nur in ein `<a href...>`-Tag einschließen. Eine solche Grafik erhält standardmäßig einen in der Farbe der Links dargestellten Rand, den man mit dem Attribut `border="0"` unterdrücken kann.

Eine Erweiterung der mit einem Link versehenen Grafik stellen die verweis-sensitiven Grafiken dar: Sie werden mit Karten versehen, die spezielle Bereiche der Grafik als Link auszeichnen: Image Maps. Diese Image Map-Technologie gibt es sowohl für die Auswertung auf dem Server als auch auf dem clientseitigen Browser. Es wird hier grundsätzlich empfohlen, client-seitige Image-Maps zu benutzen, um eine geringe Reaktionszeit und geringen Datenstrom zu erzielen.

HTML

```
<img src="url_der_Grafik" ismap usemap="#mapname">
```

XHTML

```
<img src="url_der_Grafik" ismap="ismap" usemap="#mapname" />
```

Eine derartige Grafik erhält zwei neue obligatorische Attribute:

- `ismap` (`ismap="ismap"`) gibt an, dass ein Image-Map verwendet werden soll.
- `usemap="#mapname"` gibt eine in der Datei enthaltene Karte des Namens mapname an.

Nun benötigen wir nur noch die Karte selbst. Sie wird mit dem Tag `<map>` eingeschlossen und enthält ein oder mehrere Flächendefinitionen `<area>`.

HTML:

```
<map name="mapname">
   <area shape="rect|circle|polygon|default"
      coords="koordinaten" href="url"|nohref
      [alt="beschreibung"] [accesskey="charakter"]>
   <!-- weitere Flächendefinitionen -->
</map>
```

XHTML

```
<map name="mapname">
   <area shape="rect|circle|polygon|default"
      coords="koordinaten" href="url"|nohref="nohref"
      [alt="beschreibung"] [accesskey="charakter"] />
   <!-- weitere Flächendefinitionen -->
</map>
```

- Obligatorische Attribute der Formdefinitionen:
  - `shape="form"`: Benennt die Form des anklickbaren Bereichs. Von ihr hängt die Art der ebenfalls obligatorischen Koordinaten ab.
  - `coords="koordinaten"`: Liste der zugehörigen Koordinaten.

- Formen:
  - `shape="rect"`: Rechteck, seine Koordinaten sind `coords="x1, y1, x2, y2"`. Dabei bezeichnen x_1 und y_1 die linke obere Ecke, x_2 und y_2 die rechte untere Ecke des Rechtecks. Die Koordinaten der linken oberen Ecke sind 0, 0.
  - `shape="circle"`: Kreis, seine Koordinaten sind `coords="x, y, r"`. Dabei bezeichnen x und y den Mittelpunkt des Kreises und r seinen Radius.
  - `shape="polygon"`: Polygonzug (beliebiges Vieleck), seine Koordinaten sind `coords="x1, y1, ..., xn, yn"`. Dabei bezeichnen x_1, y_1 bis x_n, y_n die Eckpunkte dieses Polygonzugs. Die ersten und letzten Koordinaten müssen übereinstimmen, damit der Polygonzug geschlossen ist.
  - `shape="default"`: Dieser Bereich besitzt keine Koordinaten und stellt das Gebiet dar, für die sonst anderweitig keine Flächendefinitionen gibt, ist also der Rest.
- `href="url"`: Bezeichnet die URL, auf die verwiesen werden soll. Anstelle der `href`-Angabe kann auch `nohref` (bzw. `nohref="nohref"`) stehen: Dann ist diese Fläche mit keinem Link verbunden.

- Optionale Parameter:
  - `alt="bezeichnung"`: In Analogie zum alt-Attribut des Bildes kann man hier jeder definierten Teilfläche ein (unterschiedliches) alt-Attribut vergeben. Diese Information wird beim Überstreichen der zugehörigen Fläche angezeigt.
  - `accesskey="charakter"`: Zeichen auf der Tastatur, das der Anwender drücken kann, um den Verweis direkt anzuspringen (access key = Zugriffstaste). Benutzen Sie nur Tasten, die auf jeder Tastatur zu finden sind. Diese Möglichkeit ist bisher jedoch kaum bei einem Browser implementiert.

4.7.2. Weitere Medienobjekte

Eingebundene Objekte werden nicht vom Browser direkt angezeigt, sondern durch sogenannte Plugins. Die Plugins sind browserabhängig. Plugins müssen extra installiert werden. Die Art des Medienobjekts wird entweder vom Server dem Browser mitgeteilt, oder die Verknüpfung zwischen Datentyp und Programm muss über die Registry bekannt sein.

Ausgaben des Plugins erscheinen in einem vom Browser verwalteten Fenster.

In Zukunft [HTML 4.0] sollen alle Fremddateien als Objekte behandelt werden.

Universelles Objekt `<object>`

```
<object data="url" type="MimeType" [declare]
   [border="breite"] [width="breite"] [height="hoehe"]
   [name="name"] [standby="Meldungstext"]
   [align="top|middle|bottom|left|right"] [shapes]>
   Alternativer Inhalt (Text oder Abbildung)
</object>
```

Die in eckigen Klammern genannten Attribute sind optional.

- Obligatorische Attribute:
  - `data`: URL-Referenz für das gewünschte Objekt.
  - `type`: Mime-Typ für das mit `data` referenzierte Objekt, wird benötigt um das richtige Plugin/Programm zu aktivieren. Mime (Multipurpose Internet Mail Extensions)-Type haben die Aufgabe, eine eindeutige Zuordnung zwischen Dateityp und Dateinamenserweiterung herzustellen. Die einfache Verbindung zwischen Suffix und Programm reicht nicht aus, weil inzwischen Suffixe für verschiedenen Dateitypen benutzt werden bzw. auf Nicht-Windows-Systemen derartiges nicht benutzt wird. Dies gilt für Browser und Server gleichermaßen.
- Optionale Attribute:
  - `declare` (bzw. bei XHTML `declare="declare"`: Objekt nur deklarieren, aber nicht ausführen. Wird meist erst z. B. durch einen Verweis vom Nutzer aktiviert.
  - `border`: Breite des Randes des Plugin-Fensters in Pixel.
  - `name`: Name des Objektes.
  - `width`: Breite des Objektfensters in Prozent oder Pixel. Angabe bei visuellen Objekten empfohlen.

- `height`: Höhe des Objektfensters in Prozent oder Pixel. Angabe bei visuellen Objekten empfohlen.
- `standby`: Meldungstext während des Ladens des Objektes.
- `align`: Ausrichtung von Beschriftungstext für das Objektfenster (`top`: Ausrichtung oben, `middle`: Ausrichtung in der Mitte, `bottom`: Ausrichtung unten. `left`: Ausrichtung links, Textfluss nach rechts; `right`: Ausrichtung rechts, Textfluss nach links).
- `text`: Text für Beschriftung des Fensters.

Das `shapes`-Attribut (XHTML `shapes="shapes"`):

Ein dem Image Map ähnlicher Mechanismus lässt sich auch bei Objekten einsetzen. Dies erfolgt aber nicht über das `<map>`-Tag, sondern über modifizierte Links (`shape`- und `coords`-Attribute):

```
<object data="hypgraph.gif" type="image/gif" shapes>
   <a href="#Anker" shape="rect" coords="1,1,249,49">Verweistext</a>
   <a href="datei.htm" shape="rect" coords="1,51,149,299">Verweistext</a>
</object>
```

Zu den einfügbaren Objekten zählen zum Beispiel Video- und Audio-Dateien, Java-Applets und ActiveX-Controls.

Spezielle Objekte: Java-Applets

1. Mit `<object>`-Tag:

```
<object classid="java:Classname" [codebase="url"] [codetype="MimeType"]>
   [<param name="name" value="wert"> ...]
</object>
```

- Obligatorisches Attribut:
  - `classid`: Name des Java-Programms ohne die Endung .class. Achten Sie auf die Einleitung mit `java:`
    Beispiel: `classid="java:animation"`.
- Optionale Attribute:
  - `codebase`: Quelle (Verzeichnis) des Java-Programms, wenn nicht im gleichen Verzeichnis wie aufrufende HTML-Seite.
    Beispiel: `codebase="../java/"`.
  - `codetype`: Typ des Programms.
    Beispiel: `codetype="application/java-vm"`.

Die für das Java-Applet nötigen Parameter können jeder für sich einzeln über das `<param>`-Tag vermittelt werden:

- Obligatorische Attribute:
  - `name`: Name eines Parameters.
  - `value`: Wert des Parameters.

2. Mit `<applet>`-Tag:

In vergleichbarer Weise lassen sich Applets seit HTML 3.2 auch mit dem `<applet>`-Tag aufrufen:

```
<applet code="url" [codebase="url"]
   [alt="Text"] [width="breite"] [height="höhe"] [hspace="x"] [vspace="y"]
   [align="left|right|top|middle|bottom"]
   [archive="url"] [mayscript]>
   [<param name="name" value="wert"> ...]
</applet>
```

- Obligatorisches Attribut:
  - `code`: Name der Class-Datei, Endung kann entfallen.
    Beispiel: `code="animation"`.
- Optionale Attribute (soweit nicht bereits beschrieben):
  - `codebase`: Quelle des Java-Programms, wenn nicht im gleichen Verzeichnis wie aufrufende HTML-Seite.
    Beispiel: `codebase="../java/"`.
  - `alt`: alternativer Text bei Nichtanzeige.
  - `archive`: URL eines ZIP-Archives, wenn Applet Teil dieses ZIP-Archives ist.
  - `mayscript`: Erlaubt JavaScript, auf das Applet einzuwirken.

- Die Nutzung des `<applet>`-Tags stößt im HTML 4.0-Standard auf Ablehnung, es sollte durch das `<object>`-Tag ersetzt werden. Wenn es schon benutzt wird, sollten die Attribute `width`, `height` und `alt` unbedingt definiert sein.

Zur Einbettung von Multimediadateien mittels `<embed>`-Tag siehe im Anhang: „Ersatz des `<embed>`-Tags“ auf S. 104.

4.8. Tabellen

Die große Bedeutung erhalten Tabellen wegen ihrer großen Vielfalt an Darstellungselementen und besonders als Tabelle mit unsichtbaren Rahmen für die relativ freie Positionierung unterschiedlicher Informationselemente sowohl in der Vertikalen als auch in der Horizontalen. Damit erobern sich Tabellen die Schlüsselstellung bei der Realisierung aufwendiger Layout-Vorstellungen.

Um es vorweg zunehmen: Es ist eigentlich nicht möglich, aber auch nicht vorgesehen, pixelgenaues Layout zu erzeugen. Dies ist auch in Hinsicht der verschiedensten Ausgabegeräte nicht sinnvoll.

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; dies ganz bei großen Tabellen durchaus länger dauern. Versuchen Sie deshalb, so viele Angaben wie möglich vorab zu definieren!

Achten Sie darauf, dass eine Tabellenzelle nur dann gezeichnet wird, wenn sie auch einen Inhalt besitzt. Eine leere Tabellenzelle sollte somit entweder einen nichtumbrechbaren Leerraum ` ` oder eine durchsichtige Grafik (Punkt) enthalten.

Wenn man im großen Maße von den Standardeinstellungen einer Tabelle abweichen möchte, bietet es sich an, die Tabellen-Definitionen (`<table>`, `<tr>`, `<td>`, `<th>`) mit Cascading Style Sheets vorzunehmen.

4.8.1. Das Tabellen-Objekt

```
<table summary="beschreibung"
   [border[="breite"]] [frame="rahmentyp"] [rules="gittertyp"]
   [cellspacing="abstand"] [cellpadding="abstand"]
   [width="breite"] [align="left|center|right"] [hspace="xbreite"]
   [vspace="ybreite"]>

   [<caption align="position">Tabellenüberschrift</caption>]

   [<tr [height="hoehe"] [align="..."] [valign="..."]>
      <th [width="breite"] [height="hoehe"] [colspan="anzahl"]
      [rowspan="anzahl"]>Kopfzelle</th>
      ...
   </tr>]
```

```
   <tr [height="hoehe"]>
      <td [width="breite"] [height="hoehe"] [colspan="anzahl"]
      [rowspan="anzahl"]>Datenzelle</td>
      ...
   </tr>
</table>
```

- Optionale Attribute des `<table>`-Tags:
  - `summary`: Angabe eines zusammenfassenden Kurztexts für die Tabelle (Zweck und Struktur der Tabelle). Von Bedeutung für die Nutzung auf nicht-visuellen Ausgabegeräten wie z.B. Braille-Zeilen. Angabe gehört zum guten Stil.
  - `border`: Randbreite in Pixeln, `border="0"` bedeutet kein Rand (siehe unten).
  - `frame`: Hiermit können Sie dann bestimmen, an welchen Seiten der Tabellenrahmen gezogen werden soll (siehe unten).
  - `cellspacing`: Breite der Gitterlinien in Pixeln.
  - `cellpadding`: Abstand des Zelleninhalts („Polsterung“) von der Gitterkante in Pixeln.
  - `width`: Breite der Tabelle oder der Zelle in Pixeln oder Prozent. Für den Fall, dass ebenfalls Breitenangaben für die Tabellenzellen vorgesehen sind, muss angemerkt werden, dass ein Mischen von Pixel- und Prozent-Werten nicht möglich ist. Die Maßeinheit der Tabellenbreite bestimmt die Maßeinheit der Zellenbreiten.
  - Ein Höhenattribut ist nicht definiert, die Tabellenhöhe wird durch die Summe aller Zeilenhöhen bestimmt.

  - `align`: Ausrichtung der Tabelle. `left`: links, Textumfluss auf rechter Seite, `center`: zentriert, kein Textumfluss, `right`: rechts, Textumfluss auf linker Seite.
  - `hspace`: horizontale Abstand in Pixeln zwischen Tabelle und Text.
  - `vspace`: vertikale Abstand in Pixeln zwischen Tabelle und Text.

4.8.2. Tabellenaußenrahmen und Gitterlinien

```
border="[breite]"
```

- Damit die Tabelle überhaupt einen Rahmen erhält, müssen Sie `border` mit angeben. Der Wert `breite` bestimmt die Dicke des Außenrahmens.
- `border="0"` bedeutet kein Rahmen.
- Einige Browser interpretieren auch die Angabe `border` ohne Wertzuweisung, was der Angabe `border="1"` entspricht.
- Im XHTML muss in jedem Fall ein Wert angegeben werden.

`frame="rahmentyp"`

- Voraussetzung für das `frame`-Attribut ist das gesetzte Attribut `border="[breite]"`.
- Mit dem Attribut `frame` können Sie dann bestimmen, an welchen Seiten der Tabellenrahmen dargestellt werden soll.
- Mögliche Werte:
  - `frame="box"`, `frame="border"`: Tabellenrahmen oben, links, rechts und unten (die Angabe ist identisch mit der Angabe `border="breite"`.
  - `frame="void"`: kein Tabellenrahmen. Wenn Sie `border="breite"` angeben, werden jedoch die Gitternetzlinien der Tabelle sichtbar angezeigt. Die Tabelle sieht dann also aus wie ein an allen Seiten offenes Gitter.
  - `frame="above"`: Rahmenlinie nur am oberen Rand der Tabelle.
  - `frame="below"`: Rahmenlinie nur am unteren Rand der Tabelle.
  - `frame="hsides"` (*engl.* horizontal sides): Rahmenlinie nur am oberen und am unteren Rand der Tabelle.
  - `frame="vsides"` (*engl.* vertical sides): Rahmenlinie nur am linken und am rechten Rand der Tabelle.
  - `frame="lhs"` (*engl.* left-hand side): Rahmenlinie nur am linken Rand der Tabelle.
  - `frame="rhs"` (*engl.* right-hand side): Rahmenlinie nur am rechten Rand der Tabelle.
- Netscape 4.x interpretiert diese Angaben nicht.

`rules="gittertyp"`

- Voraussetzung für das `rules`-Attribut ist ebenfalls das gesetzte Attribut `border="[breite]"`.
- Mit dem Attribut `rules` können Sie dann bestimmen, an welchen Stellen des Tabelleninneren Gitterlinien dargestellt werden soll.
- Mögliche Werte:
  - `rules="none"`: überhaupt keine Linien, der Außenrahmen der Tabelle wird jedoch angezeigt.
  - `rules="rows"`: Linien zwischen allen Tabellenzeilen, nicht jedoch zwischen den Spalten.
  - `rules="cols"`: Linien zwischen allen Tabellenspalten, nicht jedoch zwischen den Zeilen.
  - `rules="all"`: Linien zwischen allen Tabellenzellen (Voreinstellung).
  - `rules="groups"`: Linien zwischen Kopf, Körper und Fuß einer Tabelle (siehe unten: Erweiterte Tabellenmodelle).
- Netscape 4.x interpretiert diese Angaben nicht.

4.8.3. Tabellenüberschrift / Tabellenunterschrift

```
<caption align="position">Tabellenüberschrift</caption>
```

- Mit dem `<caption>`-Tage geben Sie die Tabellenüberschrift bzw. -unterschrift an.
- `<caption>` muss unmittelbar hinter dem `<table>`-Tag notiert werden.
- Mögliche Werte für `align` sind: `top` (Tabellenüberschrift) und `bottom` (Tabellenunterschrift). Die wohl auch möglichen Werte `left` und `right` funktionieren bei keinem Browser.

4.8.4. Die Tabellenelemente

```
<tr> ... </tr>
```

- Tag leitet eine neue Zeile der Tabelle ein.

- Optionale Attribute für die Ausrichtung von Inhalten:
  - `align` (`left|center|right|justify`): Legt die waagerechte Textausrichtung aller Zellen (!) dieser Zeile fest. Angabe des Attributes erspart die individuelle Angabe in jeder einzelnen Zelle.
  - `valign` (`top|middle|bottom|baseline`): Legt die senkrechte Textausrichtung aller Zellen (!) dieser Zeile fest. Angabe des Attributes erspart die individuelle Angabe in jeder einzelnen Zelle. Der Wert `baseline` bietet sich an, wenn die Ausrichtung anhand der Grundlinie der ersten Textzeile erfolgen soll, diese aber die den einzelnen Zellen unterschiedlich hoch ausfallen.

- Es gibt keine `width`- bzw. `hight`-Attribute! Ersteres wird durch die Tabellenbreite vorgegeben, die Höhe ergibt sich aus der größten Höhe einer Zelle dieser Reihe.

```
<th> ... </th>
```

- Tag leitet eine Zelle der Kopfzeile der Tabelle ein (Schrift standardmäßig fett und zentriert).

```
<td> ... </td>
```

- Tag leitet eine neue Zelle ein.

Beide Tags bewirken aber bis auf die unterschiedliche Schriftdarstellung dasselbe: Sie können `<th>`- und `<td>`-Zellendefinitionen beliebig mischen.

- Optionale Attribute für `<th>`- und `<td>`-Tags:
  - `width`: Breite einer Tabellenzelle. Wert wird aber nur eingehalten, wenn der Text in die somit definierte Zelle passt.
  - `height`: Mindesthöhe einer Tabellenzelle. Wert wird aber nur eingehalten, wenn der Text in die somit definierte Zelle passt. Zum anderen wird die Höhe der Zellen einer Zeile von der Zelle mit der größten minimalen Höhe bestimmt.
  - `align` (`left|center|right|justify`), `valign` (`top|middle|bottom|baseline`): Ausrichtung der Inhalte, wie oben beschrieben, aber nur die individuelle Zelle betreffend.

Die Breite wird nur realisiert, wenn Inhalt in die Tabelle bzw. Spalte entsprechend der Angabe passt, ansonst wird die Tabelle so groß wie nötig angelegt. Gesamtbreite der Tabelle hat Vorrang vor Summe der Spaltenbreiten. Die gewünschte Wirkung hat oft erst die Definition aller Spaltenbreiten.

4.8.5. Zellen verbinden

```
<td colspan="spaltenzahl" rowspan="reihenzahl"> ... </td>
<th colspan="spaltenzahl" rowspan="reihenzahl"> ... </th>
```

- Optionale Parameter:
  - `colspan`: Eine Zelle erstreckt sich über die genannte Anzahl Spalten hinweg. Die Angabe ist nur wirksam, wenn die Tabelle mindestens so viele Spalten besitzt wie angegeben.
  - `rowspan`: Eine Zelle erstreckt sich über die genannte Anzahl Reihen hinweg. Die Angabe ist nur wirksam, wenn die Tabelle mindestens so viele Reihen besitzt wie angegeben.

- Die Summe der Zellen muss in jeder Reihe bzw. Spalte der Tabelle gleich sein!

4.8.6. Tabellenfarben und -hintergründe

- Die Universalattribute `color` und `bgcolor` für Vordergrund- und Hintergrundfarben können sowohl im `<table>` als auch `<td>`- bzw. `<th>`-Tag definiert werden. Wenn die zellende-

finitionen keine Farbangaben enthalten, so übernehmen sie die Farbdefinitionen aus dem `<table>`-Tag.

- Die Farbe der Gitterlinien kann mit dem Attribut `bordercolor` im `<table>`-Tag definiert werden, ansonst wird die Farbe des Tabellenuntergrundes vom `<table>`-Tag übernommen.
- Sowohl die Tabelle im Ganzen als auch jede Zelle einzeln können mit einem Hintergrundbild ausgestattet werden: `background="URL"`. Die Angabe der Bildquelle erfolgt in derselben Weise wie für Verweise (`<a>`-Tag).

- Es bietet sich an, die Definition mit Cascading Style Sheets vorzunehmen.

4.8.7. Erweiterte Tabellenmodelle

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.

Definition von Spaltengruppen:

```
<colgroup>
   <col width="breite" />
   [<col ... >
   ... ]
</colgroup>

<colgroup width="breite" span="spaltenanzahl">
</colgroup>
```

- Optionale Parameter:
  - width: Breite der Spalte in Pixeln, Prozent oder Anteilen. Einheiten dürfen gemischt werden. Beispiele: `width="80"`, `width="10%"`, `width="2*"`. Die letzte Angabe bedeutet, dass die Spalte zwei Teile des verbleibenden Raumes breit ist; die Angabe mehrerer

Teile hat natürlich nur dann Sinn, wenn mehrere Spalten derartig definiert sind. Der gesamte Rest (also ein Teil) lässt sich auch mit `width="*"` angeben.[15]

- `span`: Wenn alle Spalten dieselbe Breite haben sollen, so lässt sich dies bereits im `<colgroup>`-Tag angeben: `span` gibt dann die gesamtanzahl der Spalten an.

- Ältere Browser (Netscape 3.x, MS Internet Explorer 3.x) interpretieren diese Angaben jedoch noch nicht. Eine sauberer Rendering beherrscht derzeit nur der Browser des Mozilla-Projekts.

Definition von logischen Tabellenbereichen:

Sie können eine Tabelle logisch in drei Bereiche aufteilen: einen Kopfbereich, einen Datenbereich und einen Fußbereich.

```
<table ... [rules="groups"]>
   <thead>
       <!-- Tabellenzeilen -->
   </thead>
   <tbody>
       <!-- Tabellenzeilen -->
   </tbody>
   <tfoot>
       <!-- Tabellenzeilen -->
   </tfoot>
</table>
```

- Anmerkungen:
  - Die Spaltenzahl und Spaltenbreiten sind in allen Bereichen identisch.
  - Die interessanteste Anwendung ist der Fall, dass Gitterlinien nur zwischen diesen Bereichen gezogen werden sollen, nicht zwischen allen Reihen, dies geschieht mit der Angabe `rules="groups"`.
  - Netscape kennt diese Angaben zumindest bis einschließlich Version 4.x nicht.
  - Wenn Sie mit Hilfe des DOM-(Document Object Model) Tabellen erschaffen, ist es sinnvoll (aber eigentlich nicht notwendig), dass diese Bereiche definiert werden. Der Internet Explorer stellt derartig erzeugte Tabellen nur unter dieser Bedingung dar!

[15] Die Definition von Frame-Breiten und -Höhen erfolgt, wie später gezeigt, in derselben Weise.

Tabellenausgabe in nicht-visuellen Medien

Für die Nutzung für nicht-visuelle Medien (z.B. Sprachausgabe) gibt es für die <td>- und <th>-Tags zwei weitere Attribute:

```
<td scope="row | col | rowgroup | colgroup">
```

scope definiert, für welche restlichen Zeilen (`col`, `colgroup`) oder Spalten (`row`, `rowgroup`) die aktuelle Zelle Überschriftencharakter hat, was z.B. für Sprachausgabe der Tabelle von Bedeutung ist.

Beispiel:

```
<table>
   <tr>
      <th scope="col">Spalte 1</th>
      <th scope="col">Spalte 2</th>
   </tr>
   <tr>
      <td>Inhalt 1a</td>
      <td>Inhalt 2a</td>
   </tr>
</table>
```

Beim Vorlesen von „Inhalt 1a“ würde immer „Spalte 1“ voran gestellt werden.

```
<td axis="Kategorientext">
```

definiert eine Liste von Kategorien, zu der die Zelle gehört, und die z.B. bei der Sprachausgabe dem Zelleninhalt voran gesprochen werden. Zum anderen lässt sich durch ein elektronisches System hiermit ermitteln, welche Tabellenzellen zu gleichen Kategorie gehören.

Beispiel:

```
<td axis="Datum">01.01.2000</td>
```

4.8.8. Anwendungsbeispiele

Horizontal- und Vertikallinien

Netscape kann keine farbigen Horizontallinien zeichnen, in keinen Fall sind jedoch Vertikallinien über ein Tag darstellbar. In diesem Fall kann man sich behelfen: Man erstellt Tabellen mit einer Zelle, deren Hintergrundfarbe die gewünschte Linienfarbe darstellt, muss die Tabellenzelle aber mit einem durchsichtigen 1-Pixel-großen Punkt füllen, damit die Zelle dargestellt wird.

Beispiele:

```
<table summary="Beispiel f&uuml;r Horizontallinie"
   bgcolor="#FF0000" width="80%" cellspacing="0"
   cellpadding="0" border="0" align="center">
<tr>
   <td><img src="dot.gif" width="1" height="1" alt="" /></td>
</tr>
</table>
```

Um bei einem textbasierenden Browser nicht gar so viel Verwirrung zu stiften, verzichten Sie auf störende Angaben wie `alt="Dies ist eine lange rote Linie"`.

Ähnlich geht es mit Horizontallinien, deren Länge, wenn nötig, aber nur in Pixeln vorgegeben werden kann:

```
<table summary="Beispiel f&uuml;r Vertikallinie"
cellspacing="0" cellpadding="0" border="0">
<tr>
   <td bgcolor="#FF0000" height="100" width="1">
      <img src="dot.gif" width="1" height="1" alt="" /></td>
   <td height="100" width="20"> </td>
   <td height="100">Noch Text daneben</td>
</tr>
</table>
```

Kombination fester und variabler Breite

Wie bereits angedeutet, ist normalerweise nicht möglich, feste (in Pixeln) und variable (in %) Spaltenbreiten zu kombinieren. Hier kommt man um das Ineinanderschachteln von Tabellen nicht umhin, die in eine äußere Tabelle variabler prozentualer Breite verschiedene Tabellen fester und/oder variabler Breite nebeneinander zu positionieren. Achten Sie darauf, dass eine

Schachtelungstiefe von drei Tabellen ineinander nicht überschritten wird. Das folgende Beispiel soll das Vorgehen demonstrieren:

```
<table summary="Tabelle mit Kombination relativer und
   absoluter Breiten" width="100%"
   cellspacing="0" cellpadding="0" border="1" align="center">
<tr align="center">
   <td>
      <table summary="Absolutbreite Tabelle" width="70"
         cellspacing="0" cellpadding="0" border="1" align="left">
      <tr>
         <td width="49"><img src="bild.gif" width="49" height="43"
            alt="Bild"></td>
         <td width="20"> </td>
         <td width="1" height="43" bgcolor="#FF0000">
           <img src="dot.gif" width="1" height="1" alt=""></td>
      </tr>
      </table>
   </td>
   <td width="100%">Zelle 2</td>
   <td>
      <table summary="Absolutbreite Tabelle" width="70"
         cellspacing="0" cellpadding="0" border="1" align="left">
      <tr>
         <td width="49"><img src="bild.gif" width="49" height="43"
            alt="Bild"></td>
         <td width="20"> </td>
         <td width="1" height="43" bgcolor="#FF0000">
           <img src="dot.gif" width="1" height="1" alt=""></td>
      </tr>
      </table>
   </td>
</tr>
</table>
```

4.9. Frames

Frames als relativ neues Element in der HTML-Syntax[16] gestattet die gleichzeitige Darstellung mehrerer HTML-Dateien auf dem Bildschirm. Als Beispiel dient hier ebenfalls das Skript-Angebot zum Workshop Internet-Präsenz.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<html>
<head>
   <title>Frames</title>

   <!-- ... -->
</head>

<frameset cols="160,*" framespacing="0" border="0"
   frameborder="no" bordercolor="#000000">
   <frame src="contents.shtml" scrolling="no" name="Contents_Frame">
   <frame src="intro.shtml" name="Main_Frame">
   <frameset ...>
   </frameset>
</frameset>

<body>
<noframes>

<h2>Notseite f&uuml;r Browser ohne Frames-Unterst&uuml;tzung</h2>

<!-- ... -->

</noframes>
</body>

</html>
```

Anhand des obigen Beispiels können Sie den Aufbau einer Definitionsdatei für Frames sehen. Das Wichtigste vorweg: Diese Seite bekommen Sie in der Regel nie zu sehen, mit dieser Seite werden sofort die in ihr angegebenen Dateien geladen. Unterstützung geben hier der textbasierte Browser Lynx bzw. Editoren wie Frontpage oder Namo.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
```

[16] Erst im HTML 4.0 standardisiert.

Beachten Sie, dass nun als Dokumenttyp `HTML 4.0 Frameset//EN` anstelle von `HTML 4.0 Transitional//EN` angegeben ist.

Unter dem Kopf folgt anstelle des `<body>`-Körpers nun die `<frameset>`-Definition.

4.9.1. Das frameset-Element

```
<frameset cols|rows="liste">
```

Für die Frameset-Definition gibt es nur ***ein*** Attribut neben den Universalattributen:

- `rows="multi_length_list"` oder `rows="multi_length_list"`. Dieses Attribut definiert ein Layout mit zeilenweise bzw. spaltenweise angeordneten Frames. Eine gleichzeitige Kombination aus spalten- und zeilenweiser Anordnung ist nicht möglich. In der Liste mit Angaben, die durch ein Komma getrennt werden, können absolute Höhen/Breiten in Pixeln, relative Höhen/Breiten in Prozent und bzw. Auffüllungen des verbleibenden Platzes:
  - `rows="150,50%,*,3*"` bedeutet:
    - Die erste Reihe ist 150 Pixel hoch.
    - Die zweite Reihe ist 50% hoch.
    - Die dritte Reihe ist ein Teil von 1+3=4 Teilen des verbleibenden Raumes hoch.
    - Die vierte Reihe ist drei Teile von 1+3=4 Teilen des verbleibenden Raumes hoch.

- Weitere proprietäre Attribute werden im HTML 4.0-Standard nicht unterstützt. Es ist aber unter Umständen durchaus sinnvoll, diese noch zu belegen, damit die frühen Browser (Netscape 3.x und höher und Internet Explorer 4.x und höher) es wie gewünscht darstellen:
  - `bordercolor="#farbe"`: Hiermit können Sie die farbige Darstellung der Rahmen zwischen den Frame-Fenstern festlegen.
  - Microsoft Internet Explorer 4.x:
    - `frameborder="1|0"`, `frameborder="yes|no"`: Legt fest, ob ein Rahmen um die Frames gezeichnet werden soll. Nach HTML 4.0-Spezifikation gehört dieses Attribut in das `<frame>`-Tag. Eine (richtige) Ausnahme gibt es: Ist das Frameset Element eines Framesets, dann gehört die Angabe hierhin!
    - `framespacing="abstand_in_pixeln"`: Gibt den Abstand zwischen den Frame-Fenstern in Pixeln an.
    - Wenn kein Rahmen gewünscht wird, müssen beide Angaben entsprechend gesetzt sein: `frameborder="0"` und `framespacing="0"`.
  - Netscape:
    - `border="breite_in_pixeln"`: Gibt die Rahmenbreite in Pixeln an. Ein Wert = 0 bedeutet keinen Rahmen.

- Wenn bei beiden Browsern kein Rahmen gewünscht wird, müssen alle Angaben entsprechend gesetzt sein: `frameborder="0"`, `framespacing="0"` und `border="0"`.

Elemente eines Framesets können Frames und Framesets sein. Damit lassen sich auch komplexe Strukturen als nur reihen- oder spaltenweise Anordnungen erreichen.

4.9.2. Das frame-Element

HTML

```
<frame src="url_oder_pfad" name="eindeutiger_fenstername"
   scrolling="yes|no|auto" noresize frameborder="1|0">
```

XHTML

```
<frame src="url_oder_pfad" name="eindeutiger_fenstername"
   scrolling="yes|no|auto" noresize="noresize" frameborder="1|0" />
```

- Unabdingbare Attribute:
  - `src="url_oder_pfad"`: Name der URL, die in dem Frame angezeigt werden soll.
  - `name="eindeutiger_fenstername"`: Name des Fensters, der Name darf nur einmal vergeben werden. Der Name darf nur Buchstaben, Zahlen und den Unterstrich „_“, aber keinen Leerraum enthalten und muss mit einem Buchstaben beginnen.
- Optionale Attribute neben den Universalattributen:
  - `noresize`: Gibt an, dass das Frame-Fenster in seiner Größe nicht verändert werden kann. Die einmalige Angabe von `noresize` genügt: Damit gilt die Eigenschaft für alle Frame-Fenster.
  - `scrolling="yes|no|auto"`: Legt fest ob und wann ein Rollbalken um das Frame-Fenster gezogen werden soll. `Yes` bedeutet Anzeige in jedem Fall, auch wenn nicht nötig. `No` bedeutet Anzeige in keinem Fall, auch wenn nötig. Eine interessante Anwendung fehlender Rollbalken ist, wenn der Designer den Eindruck vermitteln möchte, dass alle Frames zu einem einheitlichen Ganzen gehörten sollen. `Auto` bedeutet Anzeige bei Bedarf.
  - `frameborder="1|0"`: Legt fest, ob ein Rahmen um das Frame-Fenster gezeichnet werden soll oder nicht.
  - `marginwidth="breite_in_pixeln"`: Breite des linken und rechten Randes für das Frame-Fenster.
  - `marginheight="hoehe_in_pixeln"`: Höhe des oberen und unteren Randes für das Frame-Fenster.

4.9.3. Das noframes-Element

Dieses Element dient dazu, auch dann etwas auf dem Bildschirm anzuzeigen, wenn der Browser keine Frames unterstützt. Man schreibt zwischen einleitendem und schließendem Tag die übliche HTML-Syntax. Beachten Sie die Schachtelung: Das `<noframes>`-Element wird in das `<body>`-Element geschachtelt!

```
<body>
<noframes>

   <p>Alternativer Text</p>

</noframes>
</body>
```

Die Wirksamkeit des `<noframes>`-Elements lässt sich gut mit dem textbasierenden Browser Lynx überprüfen.

4.9.4. Verweise auf Frame-Fenster

Wenn Sie einen Verweis in einem Frame-Fenster vornehmen, so führt der Link dazu, dass dasjenige Frame-Fenster ausgetauscht wird, in dem der Link notiert war. Will man aber (was der Regelfall ist) den Inhalt eines anderen Frame-Fensters austauschen, so benötigt man für das Anker-Tag `<a href...>` ein weiteres Attribut: `target`.

```
<a href="url" target="ziel">
```

- `target="name_eines_frame_fensters"`: Gibt als Zielfenster das Frame-Fenster an, das den Namen `name_eines_frame_fensters` erhalten hat. Hat man einen fehlerhaften Namen angegeben (kann auch Absicht sein), so wird ein neues Browser-Fenster geöffnet.
- Einige vordefinierte Ziele bewirken besondere Reaktionen. Achten Sie darauf, dass diese Zielangaben mit einem Unterstrich beginnen:
  - `target="_blank"`: Öffnet ein neues Browser-Fenster, um die mit url angegebene Datei darzustellen.
  - `target="_top"`: Löscht das bestehende Frame und schreibt die Datei in das gesamte Anzeigefenster.
  - `target="_parent"`: Löscht das bestehende Frame und schreibt die Datei in den Zustand des Anzeigefensters, der vor dem Start des Frames aktuell war.

Mit einem Link lässt sich nur jeweils ein Frames-Fenster austauschen. Möchte man mehrere Frame-Fenster austauschen, so ist dies nur auf zwei Wegen möglich:

1. Aufbau eines neuen Framesets,
2. Nutzung von Skriptsprachen wie JavaScript, um gleichzeitig mehrere Fenster auszutauschen.

4.10. Tabellen versus Frames

Die definierte Anordnung von Informationsbausteinen sowohl in der Horizontalen als auch in der Vertikalen lässt sich (von DHTML abgesehen) nur mit Tabellen oder Frames bewerkstelligen. Eine Antwort, welcher der beiden Methoden man den Vorzug geben sollte, kann hier nicht beantwortet werden. Es sollen aber Vor- und Nachteile genannt werden:

Tabellen

- Vorteile:
  - Tabellenstrukturen werden auch von älteren Browsern verstanden.
  - Einfache Gestaltung von horizontalen und vertikalen Strukturen. Alle Strukturen werden über gemeinsame Rollbalken bewegt.
- Nachteile:
  - Es ist sehr schwer, Tabellenbestandteile mit festen und variablen Spalten zu definieren. Die Unkenntnis führt nicht selten zu „Handtuch“-Internet-Seiten und großen leeren Flächen oder als anderes Extrem, dass mein beim Lesen ständig waagerecht rollen muss.
  - Beim Druck wird alles gedruckt, nicht nur die meist mittig gesetzte Hauptinformation. Das führt auch dazu, dass das Papier nicht optimal bedruckt wird. Der Text in zu breit gesetzte Spalten wird beim Druck nicht umgebrochen, sondern abgeschnitten.
  - Einige Browser tun sich schwer bei der Anzeige tief verschachtelter Tabellen, der Ausdruck übergroßer Tabellenzellen schlägt häufig fehl.

Frames

- Vorteile:
  - Die feste Anordnung der Informationseinheiten bleibt bestehen, auch wenn in einem Teilbereich gerollt wird.
  - Einfache Definition fester und variabler Teilfenster.
  - Gute Aufteilung des Druckbildes.
- Nachteile:
  - Erst Browser der 3. Generation (Netscape 3.x, Internet Explorer 3.x).
  - Erst in HTML 4.0 standardisiert. Netscape und Internet Explorer besitzen unterschiedliche Attribute.
  - Mit Frames lassen sich aus gestalterischer Sicht nur maximal vier bis fünf Informationseinheiten auf dem Bildschirm darstellen.
  - Störende Rollbalken auf dem Desktop.

- Frames, bei denen das Rollen (Scrolling) abgeschaltet ist, lassen sich nicht rollen. So fehlen unter Umständen bei kleinen Bildschirmen Informationen.
- Den Fokus erhält immer das zuerst definierte Frame, meist links oben. Ohne den Einsatz von Skripten muss man das zu rollende oder zu druckende Teilfenster anklicken.

Einige Ratschläge

- Der sparsame, gezielte Einsatz von Frames macht diese zu einem sehr mächtigen Positionierungswerkzeug. Haupteinsatzgebiet dürften Navigationssysteme sein.
- Je kleiner eine Informationseinheit, um so eher bietet sich ein Tabelle an. Vermeiden Sie tiefe Tabellenschachtelungen (nicht mehr als drei Tabellenebenen) und Tabellen, die über die volle Dokumentlänge gehen.
- Als weitere Alternativen zu Tabellen seien Cascading Style Sheet-Definitionen zu nennen als Ersatz für die Darstellung von Rändern und frei positionierbare `<div> ... </div>`-Container.

4.11. Formulare. Vereinbarung im HTML-Text

Formulare stellen ein einfaches Mittel zur Interaktion zwischen dem Client und Server, auch wenn manchmal sehr einseitig. Die gehobene Form stellt die Auswertung durch ein CGI[17]-Skript dar, das auf dem Server läuft: Es erhält die Formulardaten und stellt eine Rückantwort dem Client zur Verfügung. Letztere Technik soll in einem separaten Kapitel behandelt werden. Nichtsdestotrotz gibt es auch einen Weg, auch ohne CGI Formulardaten an den Web-Autor zu senden: über Email. Dabei sind die Mechanismen, die für die Programmierung des Formulars in der HTML-Datei verwendet werden, in beiden Fällen gleich.

4.11.1. Der Formular-Rahmen

```
<form action="mailto:email_adresse" method="post"
   enctype="text/plain">

<!-- hier stehen u.a. die Formularelemente ... -->

</form>
```

Diese Form des Aufrufs wird bei Email-Formularen benutzt. Sie setzt allerdings voraus, dass der benutzte Browser mit einem richtig konfigurierten Email-Clienten ausgestattet ist.

Das einzige Attribut, das der Web-Autor modifizieren kann, ist das `action`-Attribut bzw. die darin enthaltene Email-Adresse. Die anderen Attribute **müssen** in jedem Fall mit angegeben sein.

Auch wenn jetzt noch nicht Gegenstand, sei bereits auf die CGI-basierte Form kurz eingegangen:

```
<form action="cgi_programm" method="post|get"
   target="fenstername">

<!-- hier stehen u.a. die Formularelemente ... -->

</form>
```

[17] Common Gateway Interface.

- Attribute:
  - `action="cgi_programm"`: URL oder Pfad des CGI-Skripts, das für die Auswertung der Daten zum Einsatz kommen soll.
  - `method="post|get"` bestimmt die Art und Weise der Datenübermittlung und Datenübergabe an das Skript-Programm. Der Web-Autor hat meist keine Entscheidungsmöglichkeit auf die zu verwendende Methode, da dies im Wesentlichen durch das Skript selbst bestimmt ist.
    - `get`-Methode: Die Nachrichten, die der Client absendet, werden an die URL angehängt. **Damit ist der Gesamtumfang inklusive der URL auf 256 Zeichen beschränkt**. Wenn die `get`-Methode zum Einsatz kommt, sollte auf das Formularelement `<textarea>` verzichtet werden, da deren Inhalt beliebig lang sein kann. Für den CGI-Programmierer bedeutet diese Methode, dass er die Formulardaten in der Umgebungsvariable `QUERY_STRING` übergeben bekommt.
    - `post`-Methode: Hier erfolgt der Versand der Formulardaten in einem separaten Datenkanal, eine Beschränkung wie bei der `get`-Methode gibt es nicht. Der CGI-Programmierer muss die Daten wie von einem Dateneingabegerät (wie z.B. die Tastatur) kommend behandeln.
    - Gute CGI-Programmierung unterstützt beide Methoden, sodass es dem Webautor überlassen bleibt, welche Methode er einsetzt.
  - `target="fenstername"` bestimmt das Frame-Fenster, in dem die Antwort ausgegeben werden soll.

Es sei hier noch darauf hingewiesen, dass der Formular-Rahmen nicht nur Formularelemente, sondern fast alle anderen HTML-Tags enthalten darf.

4.11.2. Formularelemente

Als Formularelemente können zum Einsatz kommen: Aktionsschalter, einzeilige und mehrzeilige Eingabefelder, Auswahllisten, Radiobuttons (Optionsschaltflächen) und Checkboxen (Kontrollfelder). Keins der Formularelemente besitzt ein schließendes Tag.

Formularelemente erkennen Sie in der Regel am einleitenden `<input ...>`.

Die hier vorgestellten Formularelemente werden für Sie nichts Neues darstellen. Sie werden im großen Maße in den Dialogfenstern aller graphischen Nutzeroberflächen eingesetzt, ihre Funktion und Wirkung ist plattformübergreifend standardisiert.

Alle vom Anwender modifizierbaren Elemente erhalten einen Namen, dieser muss für das Element eindeutig sein. Der Name dient der eindeutigen Zuordnung von Formularelement und eingegebenen Wert. Der Name darf nur aus den Buchstaben a–z, den Ziffern 1–9 und dem Unterstrich bestehen, er muss mit einem Buchstaben beginnen.

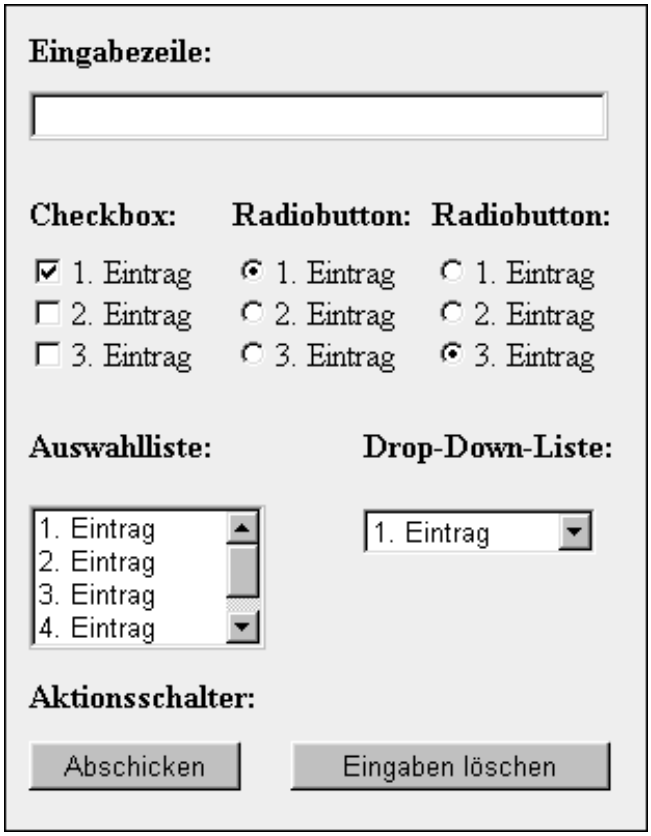

Abb. 3: Verschiedene Formen von Formularelementen

Alle Elemente können mit Event-Handlern (Prozeduren der Ereignisverarbeitung) und zugehörigen Skript-Funktionen ausgestattet sein.

- Zukünftig verfügbare optionale Parameter für alle Formularelemente:
  - `tabindex="nummer"`: Legt die Auswahlreihenfolge der Formularelemente fest, in der diese Elemente unter Nutzung der Tabulatortaste erreicht werden können. Es sind Zahlen zwischen 0 und 32767 erlaubt. Attribut kann bei `<input>`, `<textarea>`, `<select>` oder `<button>` (bzw. `<a>`) angewandt werden. Ist derzeit nur beim Internet Explorer und Mozilla implementiert.
  - `disabled="disabled"`: Zugehöriges Formularelement wird zwar angezeigt, kann aber nicht modifiziert oder selektiert werden. Attribut kann bei `<input>`, `<textarea>`,

`<select>`, `<option>`, `<optgroup>` oder `<button>` angewandt werden. Bisher nirgends implementiert.

- `accesskey="charakter"`: Zeichen auf der Tastatur, das der Anwender drücken kann, um das Formularelement direkt anzuspringen (access key = Zugriffstaste). Benutzen Sie nur Tasten, die auf jeder Tastatur zu finden sind. Diese Möglichkeit ist bisher jedoch kaum bei einem Browser implementiert. Attribut kann bei `<input>`, `<textarea>`, `<select>`, `<label>`, `<legend>` oder `<button>` angewandt werden.

Aktionsschalter

```
<input type="button" name="name" value="beschriftung"
   eventhandler="function" />
```

- Dieses Element wird hauptsächlich im Zusammenhang mit Skriptsprachen auf dem client-seitigen Browser eingesetzt.
- Obligatorische Attribute:
  - `type="button"` weist das Element als Aktionsschalter aus.
  - `name="name"` weist dem Element einen Namen zu, der für die Auswertung in einer Skript-Funktion benötigt wird. Für den Namen dürfen nur die Buchstaben a–z, die Ziffern 0–9 und der Unterstrich.
  - `value="beschriftung"` enthält die Beschriftung des Aktionsschalters.
  - `eventhandler="function"`: Als Eventhandler kommen in erster Linie die Handel `onchange`, `onclick`, `ondblclick` und `onmouseover` in Frage. `function` stellt eine Funktion dar, die in einer Skriptsprache (JavaScript, Java) geschrieben wurde.
- Die Problematik von Event-Handlern und der Sprache JavaScript wird in einem speziellen Kapitel behandelt.
- Bei der Verwendung unter XHTML muss das Tag mit einem Schrägstrich „/“ abgeschlossen sein.

Neben dem beschriebenen, relativ komplizierten, aber universell einsetzbaren Aktionsschalter sind zwei einfachere Aktionsschalter vordefiniert worden:

```
<input type="submit" value="beschriftung" />
```

- Der Aktionsschalter vom `type="submit"` bewirkt das Absenden der Daten.
- Das Attribut `value="beschriftung"` enthält die Beschriftung des Aktionsschalters.

```
<input type="reset" value="beschriftung" />
```

- Der Aktionsschalter vom `type="reset"` bewirkt das Löschen aller eingetragenen Daten.
- Das Attribut `value="beschriftung"` enthält die Beschriftung des Aktionsschalters.

Mit dem HTML 4.0-Standard dürfen Aktionsschalter auch so heißen, an Stelle von `<input>` tritt `<button>`. Implementiert ist dieses Tag im Internet Explorer ab Version 4.x und im Mozilla/ Netscape 6.x.

```
<button type="button" name="name" value="rückgabewert"
   eventhandler="function">
   Beschriftung in beliebiger Form einschließlich Bilder
</button>
```

Zu beachten ist hierbei, dass das Attribut value wie bei anderen Formularelementen nur ein Rückgabewert des Schalters ähnlich andere Formularelemente darstellt. Die Schalterbeschriftung wird zwischen öffnendes und schließendes Tag notiert: Damit können Sie auf dem Schalter alles unterbringen, was HTML/XHMTL beherrscht.

Einzeiliges Eingabefeld (einschließlich verstecktes Eingabefeld)

HTML:

```
<input type="text|password|hidden" size="laenge"
   maxlength="laenge" name="elementname"
   [value="belegung"] [readonly]>
```

XHTML:

```
<input type="text|password|hidden" size="laenge"
   maxlength="laenge" name="elementname"
   [value="belegung"] [readonly="readonly"] />
```

- Obligatorische Attribute:
  - `type="text|password|hidden"` bezeichnet die Art des einzeiligen Eingabefeldes.
    - `type="text"`: Normales einzeiliges Eingabefeld.
    - `type="password"`: Einzeiliges Eingabefeld zur Eingabe von Passwörtern. Die eingegebenen Charakter erscheinen auf dem Bildschirm als Sternchen. Es sei hier nur darauf hingewiesen, dass die Übertragung zum Server in ohne weitere Maßnahmen unverschlüsselt erfolgt!

    - `type="hidden"`: Versteckte Eingabezeile. Die enthaltenen Daten werden aber übermittelt. Typischer Einsatzzweck ist die erneute Ausgabe fehlerhaft ausgefüllter Formulare, wobei die richtigen Angaben „versteckt“ werden.
  - `size="laenge"`: Länge des Eingabefeldes in (durchschnittlichen) Buchstabenbreiten.
  - `maxlength="laenge"`: Maximale Anzahl der im Eingabefeld eingebbaren Charaktern. Wenn der Wert für `size` kleiner ist als der für `maxlength`, so wird die Eingabe im Feld gerollt.
  - `name="elementname"`: Für die Auswertung benötigte Angabe des Formularelements (Hier nur Buchstaben a–z, Ziffern 1–9 und den Unterstrich eingeben).
- Optionale Attribute:
  - `value="belegung"`: Anfangsbelegung der Eingabezeile.
  - `readonly`: Eine Modifikation des Eingabefeldes ist nicht möglich.

Mehrzeiliges Eingabefeld

HTML:

```
<textarea cols="anzahl" rows="anzahl"
   name="elementname"
   [value="belegung"] [readonly]></textarea>
```

XHTML:

```
<textarea cols="anzahl" rows="anzahl"
   name="elementname"
   [value="belegung"] [readonly="readonly"]></textarea>
```

Das End-Tag `</textarea>` ist nötig und darf nicht weggelassen werden! Diese Schreibweise gilt auch für XHTML, eine Abkürzung der Form `<textarea ... />` ist unzulässig.

Die Angaben `rows=` und `cols=` bestimmen lediglich die Anzeigegröße des Eingabebereichs, nicht die Länge des erlaubten Textes. Die ist theoretisch unbegrenzt.

- Obligatorische Attribute:
  - `cols="anzahl"`: Anzahl der Textspalten, bezogen auf einen diktengleichen Font.
  - `rows="anzahl"`: Anzahl der Textzeilen.
  - `name="elementname"`: Für die Auswertung benötigte Angabe des Formularelements (Hier nur Buchstaben a–z, Ziffern 1–9 und den Unterstrich eingeben).

- Optionale Attribute:
  - `value="belegung"`: Anfangsbelegung des mehrzeiligen Eingabefeldes.
  - `readonly`: Eine Modifikation des Eingabefeldes ist nicht möglich.

- Die Textlänge im `<textarea>`-Formularelement ist nicht beschränkt. Benutzen Sie im Zusammenhang mit diesem Formularelement nur die Formularmethode `method="post"`!

Auswahllisten

```
<select name="elementname" size="anzeigegroesse" multiple>
   <!-- Listenelemente -->
</select>
```

- Obligatorische Attribute:
  - `name="elementname"`: Für die Auswertung benötigte Angabe des Formularelements (Hier nur Buchstaben a–z, Ziffern 1–9 und den Unterstrich eingeben).
  - `size`: Anzeigegröße der Liste (Zeilenanzahl). Wenn Sie `size="1"` angeben, definieren Sie eine sogenannte „Drop-Down-Liste“.
- Optionales Attribut:
  - `multiple`: gestattet Mehrfachauswahlen.
- Obligatorisch ist auch mindestens ein Listenelement, das mit `<option>` definiert wird.

HTML:

```
<option value="wert_n" [selected]>
```

XHTML:

```
<option value="wert_n" [selected="selected"] />
```

- Obligatorische Attribute:
  - `value="wert_n"`: Rückgabewert der Auswahl. Achten Sie darauf, dass sich die Rückgabewerte der einzelnen Listeneinträge unterscheiden.
- Fakultative Attribute:
  - `selected`: Vorgegebene Standardauswahl.

Auswahl-Schaltflächen

Radiobuttons (Optionsschaltflächen, d.h. nur eine Auswahl möglich)

HTML:

```
<input type="radio" name="elementname" value="wert" checked>
```

XHTML:

```
<input type="radio" name="elementname" value="wert" checked="ckecked" />
```

- Aussehen: Kreisrunde Felder, die einzige Auswahl ist durch einen Punkt im Kreis gekennzeichnet. Sinnvollerweise sollten mindestens zwei derartige Formularelemente pro Gruppe angegeben werden.
- Obligatorische Attribute:
  - `name="elementname"`: Für die Auswertung benötigte Angabe des Formularelements (Hier nur Buchstaben a–z, Ziffern 1–9 und den Unterstrich eingeben). Alle Radiobuttons, **die den gleichen Namen haben**, gehören zu einer Gruppe, d.h., von diesen Buttons kann der Anwender genau einen markieren.
  - `value="wert"`: Rückgabewert.
- Fakultative Attribute:
  - `checked`: Definiert die Standard-Vorgabe.

Checkboxes (Auswahlfelder, d.h. mehrere Angaben möglich)

HTML:

```
<input type="checkbox" name="elementname" value="wert" checked>
```

XHTML:

```
<input type="checkbox" name="elementname" value="wert" checked="checked" />
```

- Aussehen: Quadratisches Felder, die mögliche Auswahl ist durch ein Kreuz oder durch ein Häkchen im Quadrat gekennzeichnet.
- Wenn mehrere Felder einer Gruppe ausgewählt sind, wird jeder Elementwert einzeln zurückgegeben, die Einzelwerte werden in keiner Form zusammengefasst.

- Obligatorische Attribute:
  - `name="elementname"`: Für die Auswertung benötigte Angabe des Formularelements (Hier nur Buchstaben a–z, Ziffern 1–9 und den Unterstrich eingeben). Alle Checkboxen, die den gleichen Namen haben, gehören zu einer Gruppe, d.h. von diesen Elementen kann der Anwender keines, eines oder mehrere ankreuzen.
  - `value="wert"`: Rückgabewert.
- Fakultative Attribute:
  - `checked`: Definiert die Standard-Vorgabe.

Verstecktes Element

```
<input type="hidden" name="name" value="beschriftung" />
```

Sie können Felder in einem Formular definieren, die dem Anwender nicht angezeigt werden. Versteckte Felder können Daten enthalten. Beim Absenden des Formulars werden die Daten versteckter Felder mit übertragen. Auf diese Weise können Sie beispielsweise zusätzliche Informationen an CGI-Programme übergeben oder erläuternden Text einfügen, der bei der E-Mail-Übertragung der Formulardaten in der E-Mail mit enthalten ist.

4.11.3. Label und Gruppierungen

Bisher stehen die Formularelemente allein im Raum. In Dialogfenstern gibt es aber weitere Elemente, die auch in Formularen einsetzbar sind:

Label

```
<label>
   <formularelement mit name="elementname"> labelname
</label>
```

oder

```
<label for="elementname">labelname</label>
<formularelement mit name="elementname">
```

Hiermit wird das angegebene Formularelement mit dem in `labelname` angegebenen Beschreibungstext versehen. Auch dieser Text ist mit der Maus anklickbar.

Gruppierung

Hinter diesem Mechanismus verbirgt sich ein Verfahren, mehrere Feldelemente als zusammengehörig mit `<fieldset>` zu kennzeichnen (häufig mit gemeinsamen Rahmen versehen) und dieser Gruppe eine optionale gemeinsame Beschreibung mit `<legend>` zu geben.

```
<fieldset>
   <legend align="ausrichtung">gruppenbezeichnung</legend>

   <feldelemente evtl. mit Attribut disabled>

</fieldset>
```

Das zusätzliche Attribut `disabled`, das allen Formularelementen zugeordnet werden kann, zeichnet dieses Element als nicht zur Gruppe zugehörig aus.

5. Eintragungen im HTML-Kopf

5.1. Titel

Der Titel wird in der Regel in der Programmzeile des Browsers angezeigt. Bitte wählen Sie einen möglichst aussagekräftigen Titel, vermeiden Sie Titel wie „Willkommen auf meiner Homepage“ oder “Unknown Title”.

Die Länge des Titels sollte 60 Charakter nicht übersteigen.

5.2. Meta-Tags

5.2.1. Einführung

Meta-Tags werden immer im HTML-Kopf notiert.

Es gibt zwei verschiedene Sorten von Meta-Tags:

- Tags mit Informationen, die auch über das HTTP-Protokoll ausgetauscht werden könnten. Häufig ist es aber nicht möglich, auf sie zu verzichten, z.B. weil dem Autoren die Zugriffsrechte für den Server fehlen oder weil es nicht möglich ist, für alle HTML-Dateien gleiche HTTP-Parameter wie z.B. Sprache des Dokuments zu übertragen.
- Tags mit Angaben für Suchmaschinen.

Die Definitionen sind leider nicht sehr einheitlich, und auch vorhandene Vorschläge sind noch nicht in einem Zustand, dass sie standardisiert und allgemein anerkannt sind. So wird es hier nur eine kleine Auswahl wichtiger und seit längerem im Einsatz bewehrter Attribute und Inhalte geben.

Meta-Tags besitzen keine schließenden Tags. Im Falle von XHTML müssen sie mit einem Schrägstrich abgeschlossen werden.

5.2.2. Tags mit Informationen aus dem HTTP-Protokoll (http-equiv)

Aufbau des Tags:

- `<meta http-equiv="name" content="inhalt" />`

```
<meta http-equiv="Content-Type" content="zeichensatz" />
```

- Hiermit legen Sie den zu verwendeten Zeichensatz fest. Achten Sie auf die genaue Schreibung: Vergessen Sie nicht das „text/html“
- Beispiele:
  - `<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />`
  - `<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />`
- Beachten Sie ebenfalls, dass die Angabe des Zeichensatzes nicht in Anführungszeichen steht.

```
<meta http-equiv="Content-Language" content="language" />
```

- Sprache des Dateiinhalts. Z.B. `content="de"`, `content="en-en"`.

```
<meta http-equiv="refresh" content="zeit_in_sec" url="url" />
```

- Diese Anweisung bewirkt, dass nach der angegebenen Zeit in Sekunden die nachstehend genannte HTML-Seite nachgeladen wird und die bestehende Seite ersetzt. Beachten Sie, dass dies nicht alle Browser können, geben Sie Ihrem Leser die Chance einer manuellen Weiterleitung, d.h. binden Sie auf der ersten Seite einen entsprechenden Link ein.
- Die URL muss nicht mit dem eigenen Server identisch sein. Einer der Anwendungszwecke ist z.B. das Weiterleiten von Anfragen bei umgezogenen Inhalten.

```
<meta http-equiv="expires" content="zeitpunkt" />
```

- Zeigt das „Verfallsdatum“ der Datei an. Proxy-Servern und dem Browser wird damit angezeigt, wann er die Datei erneuern muss. Dies verhindert, dass veraltete Dateien aus Zwischenspeichern anstelle der Originaldatei verwendet werden.
- Beispiele für Zeitpunkte:
  - `content="0"`: In jedem Fall Seite vom Ursprungsserver holen.
  - `content="Sat, 13 May 2000, 12:00:00 GMT"`: Verfall zum angegebenen Termin. Notieren Sie die Angabe wie im Beispiel. Als Wochentagkürzel sind erlaubt: `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, `Sat` und `Sun`. Als Monatsnamen sind erlaubt: `Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`, `Nov` und `Dec`.
  - `content="zeit_in_sec"`: Bewirkt das Nachladen der Datei nach Ablauf der angegebenen Zeit in Sekunden.

```
<meta http-equiv="pragma" content="no-cache" />
```

- Bewirkt, dass die entsprechende Datei nicht auf einem Proxy-Server gespeichert werden soll.

5.2.3. Angaben für Suchmaschinen (name)

Wir stellen Ihnen hier bewusst nur wenige Varianten vor: Einen endgültigen Standard gibt es hierfür noch nicht.

Aufbau des Tags:

- `<meta name="name" content="inhalt" />`

`<meta name="description" content="inhalt" />`

- Enthält eine kurze Inhaltsangabe, die die Suchmaschine anstelle der ersten Wörter des Dokuments ausgeben soll.
- Der Inhalt sollte 150 Zeichen (inklusive Leerräume) nicht übersteigen.

`<meta name="keywords" content="liste" />`

- Enthält eine Liste von Stichwörtern, die die Suchmaschine auswerten kann. Dies bietet sich vor allem dann an, wenn das Stichwort als Wort in der Datei nicht vorkommt.
- Der Inhalt sollte 870 Zeichen (inklusive Leerräume und Kommas) nicht übersteigen.
- Trennen Sie die Stichwörter mit Kommas, lassen Sie hinter jedem Komma einen Leerraum.
- Es gibt Stichwörter, die keine Suchmaschine Ernst nimmt: Sex, Microsoft...

`<meta name="autor" content="autor_name" />`

- Name des Autors.

`<meta name="revisit-after" content="zeitraum" />`

- Angabe an den Such-Robot, nach wie viel Tagen er wieder vorbeikommen soll.

`<meta name="robots" content="befugnisse" />`

- Hiermit kann der Suchmaschine mitgeteilt werden, wie mit der Seite zu verfahren ist:
- Einzelbefugnisse:
  - Erstes Befugnis: `content="index"`: Seite soll von der Suchmaschine in ihre Datenbank aufgenommen werden. `content="noindex"`: Seite soll *nicht* von der Suchmaschine in ihre Datenbank aufgenommen werden.
  - Zweites Befugnis: `content="follow"`: Die Links auf der Seite sollen weiterverfolgt werden. `content="nofollow"`: Die Links auf der Seite sollen nicht weiterverfolgt werden.
- Die beiden Einzelbefugnisse können beliebig (vier Möglichkeiten) kombiniert werden.
- Vereinfachte Angaben:
  - `content="all"` bedeutet: `content="index"` *und* `content="follow"`.

  - `content="none"` bedeutet: `content="noindex"` *und* `content="nofollow"`.
- Eine fehlende Tag-Angabe bedeutet im Interesse der Suchmaschine `content="all"`.

5.2.4. Was Sie sich schenken können

- Suchmaschinen werten nur die erste Titelangabe aus. Mehrere (gleiche) Titel bringen nichts.
- Angaben von Stichwörtern, die mit der Seite nichts zu tun haben. Massenstichwörter wie Sex oder Microsoft führen unter Umständen zur Herunterstufung der Bedeutung der Seite oder zu einem Ignorieren Ihrer Seite.
- Jedes Schlüsselwort (keyword) wird nur einmal aufgenommen. Stichwörter in Ein- und Mehrzahl sind aber verschiedene Stichwörter.
- Mehrfach-Anmeldung bei einer Suchmaschine. Wenn Sie gefunden wurden, dann bekommen Sie ohnehin wieder Besuch.

5.3. Link-Tags

Link-Tags werden immer im HTML-Kopf notiert. Sie besitzen kein schließendes Tag.

Diese gibt es ebenfalls in zwei Formen:
- Angabe von Beziehungen zu anderen Dateien.
- Bezug von Cascading Style-Sheet-Dateien.

- `<link rel="art_des_bezugs" href="url_oder_pfad" title="titel" />`
  - `titel="titel"` enthält eine erläuternde Angabe.
  - `href="url_oder_pfad"`: URL- oder Pfad-Angabe in der bekannten Weise.
  - Bezug-Angaben:
    - `rel="contents"` steht für einen Bezug zur Inhaltsverzeichnis-Datei.
    - `rel="chapter"` steht für einen Bezug zur Kapitel-Datei.
    - `rel="section"` steht für einen Bezug zum Abschnitt.
    - `rel="subsection"` steht für einen Bezug zum Unterabschnitt.
    - `rel="index"` steht für einen Bezug zur Stichwortverzeichnis-Datei.
    - `rel="glossary"` steht für einen Bezug zur Glossar-Datei.
    - `rel="appendix"` steht für einen Bezug zum Anhang.
    - `rel="next"` steht für einen Bezug zur nächsten, d.h. logisch folgenden Datei.
    - `rel="prev"` steht für einen Bezug zur vorherigen, d.h. logisch vorangegangenen Datei.
    - `rel="start"` steht für einen Bezug zur ersten Datei.
    - `rel="help"` steht für einen Bezug zu einer Datei mit Hilfestellung.
    - `rel="bookmark"` steht für einen Bezug einem Orientierungspunkt mit Lesezeichen (bookmarks).
    - `rel="alternate"` steht für einen Bezug zu einer Datei mit gleichem Inhalt.

- `<link rel="stylesheet" type="text/css" href="url_oder_pfad" title="titel" />`
  - Bezug zu einer Datei mit Style Sheet-Angaben.

- Unter XHTML wird der Bezug zu einer Style Sheet-Datei wie folgt notiert:
  - `<?xml-stylesheet type="text/css" href="url_oder_pfad" ?>`
  - Im Interesse der Abwärtskompatibilität sollten in einem XHTML-Dokument beide Angaben (gleichartig) vorgenommen werden.

5.4. Weitere mögliche Einträge im HTML-Kopf

- Kommentare.
- Skripten (z.B. JavaScript)
- Stildefinitionen.

6. Going International: Zeichensätze und UNICODE

6.1. Etwas konfus: Unterschied Charakter und Glyph

Der Begriff Charakter[18] bezieht sich auf „kodierte Buchstabendarstellung“, der letztendlich nichts anderes als einen Kodewert darstellt. Als Glyph[19] hingegen wollen wir dagegen die Darstellung des Charakters auf dem Bildschirm oder auf Papier mit seiner spezifischen physischen Form (das wäre z.B. ein Schriftfont) und Größe verstehen.

In der UNICODE-Terminologie gibt es noch den Begriff des „abstrakten Charakters“, das ist ein Charakter als Element eines Kode-Repertoires. Nun müßte es einen Unterschied zwischen abstrakten und konkreten Charakter, nämlich seiner Darstellung z.B. auf dem Bildschirm, geben. Aber das abstrakte Charakterkonzept bei UNICODE geht wohl über die reine Kode-Fixierung hinaus. Glyph ist dann die tatsächliche physische Form eines abstrakten Charakters mit spezifischer Form und Größe. Nun ist doch alles klar!?

6.2. Internationale Zeichensätze

Zeichensatzkodierungen gibt es nicht erst seit dem Aufkommen von Computern. Eines des frühen Systeme stellt z.B. das internationale Telegraphen-Alphabet Nr. 2 (CCITT-Code Nr. 2) dar, das den Tasten eines Fernschreibers eine Kodierung, nämlich als serielle Abfolge von Strom- und Kein-Strom-Schritten definierte. Aus heutiger Sicht stellt es ein Fünf-Bit[20]-System dar, d.h. 32 Tasten waren definierbar. Dazu gehören unter anderem auch Codes für *Line Feed* (Zeilenschaltung) und *Carriage Return* (Wagenrücklauf). Eine Taste diente einem besonderen Mechanismus: dem unsäglichen *Escape*-Mechanismus. D.h., man entweicht aus dem aktuellen Bereich darstellbarer Charakter in einen anders gearteten. So gab es denn auch Fernschreiber, die lateinische und kyrillische bzw. lateinische und arabische[21] Schriften beherrschten.

---

[18] Hier als Überbegriff auch für Gesamtheit aller Buchstaben und Zeichen.

[19] *engl.* Skulptur, Gestalt

[20] Bit stellt eine Stelle des auf Computern angewandten binären Zahlensystems, ein Zahlensystem zu Basis 2, dar. Jede Stelle kann zwei Werte annehmen: nämlich 0 und 1. Wenn man den Bereich darstellbarer Zahlen vergrößern möchte, schreibt man weitere Ziffern davor: mit zwei weiteren Stellen erreicht man so z.B. 2×2×2=8 (ähnlich zum Zehnersystem 10×10×10=1000) Möglichkeiten. Alle Zahlensysteme sind untereinander eineindeutig umrechenbar.

[21] Das waren kleine Wunderwerke deutscher Feinmechanik: Diese Maschinen kannten nicht nur die umgekehrte Schreibrichtung, sondern auch halbe Buchstabenbreiten. Versuchen Sie dies mal mit Ihrem Microsoft-Betriebssystem!

Mit dem Aufkommen von Rechenmaschinen wurde der Wertebereich auf sieben Bit vergrößert, der zugehörige Wertebereich betrug demgemäß 0–127, der zugehörige Standard sollte später unter ASCII[22]-Standard bekannt werden. Der ASCII-Standard ist aber nur eine nationale Variante (nämlich die amerikanische) eines internationalen Standards, der allerdings Platzhalter enthielt. So wurden in der amerikanischen Variante diese Plätze mit `[|]<~>` belegt, während die deutsche Variante `ÄÖÜäöü` bekam.

Mit dem Aufkommen (amerikanischer) Computer war mit dem ASCII-Standard nicht mehr viel zu machen, der ASCII-Zeichensatz wurde nochmals um ein Bit zum einem vollständigen Byte[23] erweitert. Die nationalen Versionen der Betriebssystem wie DOS und Windows wurden dann mit national angepassten Zeichensätzen (durch so genannte Code-Pages realisiert) ausgestattet. Mit der Einführung von Windows hat man dann auf den ANSI-Standard[24] gesetzt, der einen Großteil der in der westlichen Welt gebräuchlichen Schriftzeichen abdeckt. Nichtsdestotrotz war man auf 8-Bit-Basis angewiesen, weitere nationale Standards zu definieren und einzusetzen, die in der ISO[25] 8859-1 definiert sind.

[22] American Standard Code for Information Interchange.

[23] Das Binärsystem ist auf Dauer etwas gewöhnungsbedürftig. So hat man jeweils 4 oder 8 Bit sowohl hardware- als auch zahlenmäßig zu einem Halb-Byte bzw Byte (Oktett) zusammengefasst. Die Prozessorbefehle können somit auch 4, 8, 16 oder 32 Bit in einem Schritt verarbeitet. Was die Zahlenangabe anbetrifft, so erfolgt diese nun ebenfalls in 4 Bits (8 Bit = 256 Ziffern wäre auch nicht das Gelbe vom Ei). 4 Bits = 16 Möglichkeiten bzw. Ziffern lassen sich relativ gut vorstellen; für die „Ziffern“ 10–15 werden die ersten Buchstaben des Alphabets a–f benutzt. Das so genannte Hexadezimalsystem lässt sich eineindeutig in das Dezimal- oder das Binärsystem umrechnen.

[24] Benannt nach dem US-amerikanischen Normungsinstitut American National Standards Institute.

[25] International Standardization Organization.

Definierte Alphabete

Standard	Name des Alphabets	Beschreibung
ISO 8859-1	Lateinisches Alphabet Nr. 1	Western, westeuropäisch
ISO 8859-2	Lateinisches Alphabet Nr. 2	zentraleuropäisch
ISO 8859-3	Lateinisches Alphabet Nr. 3	Südeuropäisch, Maltesisch, Esperanto
ISO 8859-4	Lateinisches Alphabet Nr. 4	Nordeuropäisch
ISO 8859-5	Lateinisch-kyrillisches Alphabet	Slawische Sprachen
ISO 8859-6	Lateinisch-arabisches Alphabet	Arabische Sprachen, Farsi
ISO 8859-7	Lateinisch-griechisches Alphabet	Modernes Griechisch
ISO 8859-8	Lateinisch-hebräisches Alphabet	Hebräisch und Jiddisch
ISO 8859-9	Lateinisches Alphabet Nr. 5	Türkisch
ISO 8859-10	Lateinisches Alphabet Nr. 6	Nordisch (Sámi, Inuit, Isländisch)
ISO 8859-11	Lateinisch-Thai Alphabet	Thai
ISO 8859-13[26]	Lateinisches Alphabet Nr. 7	Baltische Sprachen
ISO 8859-14	Lateinisches Alphabet Nr. 8	Keltische Sprachen
ISO 8859-15	Lateinisches Alphabet Nr. 9	Euro

ISO Latin 9 unterscheidet sich vom stark benutzten ISO Latin 1 nur in wenigen Zeichenpositionen. Dies betrifft im Wesentlichen des Euro-Währungssymbol €, š, Š, ž, Ž, œ, Œ und Ÿ.

Aber auch hiermit ist auf Dauer kein Dokumentenaustausch möglich. Eine Erweiterung des Kodeumfanges auf zwei oder vier Byte ist daher unumgänglich, wie dies bei einer Reihe von Zeichensätzen aus dem CJK[27]-Raum bereits der Fall ist.

[26] Teil 12 der ISO 8859 nicht definiert.

[27] China, Japan, Korea.

6.3. UNICODE (ISO 10646)

- Literatur:
  - UNICODE-Konsortium: http://www.unicode.org/

Das UNICODE-Konsortium hat mit der Neudefinition eines universellen Charakterzeichensatzes UCS[28] das Ziel verfolgt, einen plattformunabhängigen, programmunabhängigen und sprachenunabhängigen Zeichencode bereitzustellen. Jeder Charakter wird dabei mit zwei Bytes (UTF-16[29]) oder vier Bytes (UTF-32) dargestellt.

Obwohl sich die Definition noch im vollem Gange befindet, sind Teile bereits standardisiert und teilweise in Betriebssystemen und Anwendungsprogrammen implementiert.

Hierzu gehören:
- Microsoft Windows 2000, AIX, Sun, HP/UX.
- HTML, XML. Netscape ab 4.0, Internet Explorer ab 5.0.
- Datenbanken: Sybase, Oracle, DB2.
- Textverarbeitungsprogramme: Microsoft Office 2000, WordPerfect 8 (Import und Export), WordPerfect 2000.

Allerdings sollte sich herausstellen, dass auch 65536 Code-Varianten nicht ausreichen, weitere Ebenen (planes) können für weniger gebräuchliche Zeichensätze definiert werden. Diese zusätzlichen Ebenen können über zwei verschiedene Mechanismen erreicht werden:
- So genannte *surrogate pairs*, damit können im 2-Byte-Bereich von 0xD800–0xDFFF derartige Zeichen etwas umständlich zwar, aber immerhin kodiert werden. Auf diese Weise ist der Zugang zu über 1 Million Charakter möglich. Die umständliche Kodierung wird nur bei weniger gebräuchlichen Charakter notwendig.
- Die Erweiterung auf 4 Byte (UTF-32). Damit ist jeder Charakter einfach kodierbar, allerdings steigt der Speicheraufwand auf das Doppelte. Zum anderen ist gegenwärtig ungewiß, wann erste SoftwareProdukte dies unterstützen.

Gegenwärtig sind bereits
- UNICODE Version 2.1: 38887 Charakter und 47402 16-Bit-Werte kodiert bzw.
- UNICODE Version 3.0: 49194 Charakter und 57709 16-Bit-Werte kodiert.

[28] Universal Character Set

[29] Uniform transformation format.

Zu den kodierten Werten gehörten:

- Charakter (Buchstaben und Zeichen),
- kombinierbare diakritische Zeichen und Markierungen in voller und halber Breite,
- Steuerzeichen (u.a. in der Tabelle *General Punctuation* (0x2000–0x206F), unter anderem für Bidirektionalität,
- Räume für *surrogate pairs* und private Zeichensätze.

Eine Besonderheit stellen die kombinierbaren Zeichen dar. Ein beliebiger Charakter kann mit einem oder mehreren dieser kombinierbaren Zeichen an einer Textposition ausgegeben werden: z.B. w̌ bestehend aus w und dem *háček*-Akzent ˇ. Dazu schreiben Sie den Grundbuchstaben und lassen ihn von beliebig vielen kombinierbaren Zeichen folgen. Es sei hier aber noch angemerkt, dass der Charakter ñ für den Rechner nicht identisch mit der Kombination aus n und ˜ ist!

Formen der Kodierung:

UTF-16 Standard-UNICODE-Kodierung in zwei Bytes für jeden Charakter.

UTF-32 UNICODE-Kodierung in vier Bytes für jeden Charakter. Der Mechanismus der *surrogate pairs* ist hiermit nicht mehr notwendig.

UTF-8 Byteweise Kodierung. Charaktercodes kleines als 128 werden so kodiert wie sie sind. Dies betrifft im Wesentlichen das ASCII-Repertoire. All diese Charakter besitzen als führendes Bit eine 0. Alle anderen UNICODE-Zeichen werden in einem sehr komplizierten Verfahren in eine Abfolge von zwei bis sechs Bytes, jeweils mit einem führenden „1“-Bit versehen. Diese Transformation von UTF-16 in UTF-8 ist eineindeutig, d.h., wechselseitig sind Angaben in die jeweils andere Form der Kodierung umsetzbar. Als Alternative steht die Angabe über Nummerkodierungen der Form &#...; oder als Entity, da die hierfür benötigten Codes im Bereich unter 128 liegen. **Achtung**: Die aus den ISO 8859-Zeichensätzen bekannten Codes größer 127 müssen ebenfalls UNICODE-kodiert werden!

UTF-7 Byteweise Kodierung. Jeder Charaktercode wird als Folge von einem oder mehr Bytes im Wertebereich von 0 bis 127 umgesetzt. Damit ist die Übertragung als 7-Bit-Bytes, dem ursprünglichen Internet-Standard möglich. Ein großer Teil der ASCII-Zeichen kann direkt dargestellt werden, allerdings werden einige Werte für Escape-Bytes, die eine Erweiterung einleiten, reserviert.

Die Konvertierung der o.g. in das jeweils andere Transformationssystem ist eindeutig möglich.

6.4. Vereinbarung des gewünschten Zeichensatzes

Der gewünschte Zeichensatz wird in einer HTML-Datei im HTML-Kopf als Meta-Tag vereinbart:

```
<meta http-equiv="Content-Type" content="zeichensatz" />
```

- Beispiele:
  - `<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />`
  - `<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />`

Achten Sie auf die genaue Schreibung: Vergessen Sie nicht das „text/html“. Beachten Sie ebenfalls, dass die Angabe des Zeichensatzes nicht in Anführungszeichen steht.

Im XHTML-Dokument wird der Zeichensatz in der XML-Deklaration angegeben:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
```

Auch hier gilt, dass im Interesse der Abwärtskompatibilität beide Angaben vorgenommen werden sollten.

7. Regeln XHTML-konformer HTML-Syntax

- `<html>`-, `<head>`- und `<body>`-Elemente sind notwendig.
- `<title>`-Tag muss der erste Eintrag im `<head>`-Element sein.
- Achten Sie auf Kleinschreibung von Tag- und Attribut-Namen.
- Werte für Attribute müssen in Anführungszeichen eingeschlossen werden.
  - Beispiel: `align="right"`.
- Achten Sie auf fehlerfreie Verschachtelung (Nesting) von Tags.
  - Beispiel `<b><i>Text</i></b>`.
- Jedes Tag muss mit einem schließenden Tag versehen werden.
  - Beispiel: `<li>Listeneintrag</li>`.
- Bei leeren Tages (d.h. solchen, die bisher kein schließendes Tag besaßen) muss ebenfalls ein schließendes Tag angegeben werden. Hier kann man aber eine verkürzte Schreibweise anwenden:
  - Beispiel: `<hr />`

  Wenn man zwischen dem Tag-Namen und dem Schrägstrich einen Leerraum belässt, so wird dies auch von den bisherigen Browsern verkraftet: „/“ ist ein Attribut, das sie nicht kennen und somit nicht ausführen.
- Tags, die im HTML 3.2 ohne schließendes Tag benutzt wurden: `<br>`, `<frame>`, `<hr>`, `<img>`, `<input>`, `<link>`, `<meta>`, [`<wbr>`].
- Es gibt keine leeren Attribute mehr, d.h., jedes Attribut muss einen Wert zugewiesen bekommen:
  - Beispiele:
    - `align="right"`,
    - `noshade="noshade"`.
- Die Bezeichnung von Ankern erfolgt über `id="name"`, nicht mehr über `name="name"`. Um Cross-Browser-Kompatibilität zu erreichen, sollte beides angegeben werden.

- Veränderte Schreibweise bei Skripten:
  - Beispiel:

```
<script ...>
   <![cdata[
      Jetzt folgt das Skript
   ]]>
</script>
```

8. Hinweise zur nutzer- und behindertenfreundlichen Gestaltung von Web-Seiten

- Das `<img>`-Tag benötigt in jedem Fall das `alt`-Attribut zur Erläuterung des Bildinhaltes. Wenn der Platz nicht ausreichen sollte, so benutzen Sie das `longdesc`-Attribut, so dass Sie hiermit eine umfangreichere Beschreibung verfügbar machen können.
  Beispiel: `<img ... longdesc="url">`
  Momentan wird diese Möglichkeit nur von Mozilla unterstützt, der Link wird ausgegeben, wenn man mit der rechten Maustaste auf das entsprechende Bild drückt.
  Wenn eine übergreifende Lösung gewünscht wird, bietet sich ein Link z.B. in der Bildunterschrift an.

- Aufeinander folgende Links sollten nicht nur durch einen Leerraum oder einen Zeilenumbruch getrennt sein.

- Versehen Sie das Link-Tag `<a>` mit dem `title`-Attribut. Bei Überfahren des Links mit der Maus erhält der Betrachter diese Information in einem Tool-Tip-Fenster.

- Sie können in die Auswahl-Reihenfolge der Links bzw. Formularelemente, die über die Tabulator-Taste erreicht werden sollen, mit dem Attribut `tabindex="nummer"` eingreifen. Die Nummer legt die Auswahlposition fest.

- Links und Formularelemente lassen sich mittels des `accesskey`-Attributs leicht über Tastatur ansteuern. Es wird empfohlen, nach Möglichkeit nur solche Buchstaben auszuwählen, die nicht für den Browser verwendet werden. Letzteres ist sehr schwer, mit der Tabulator-Taste lässt sich aber auch zwischen URL und Brwoserfenster einfach wechseln.

- Verwenden Sie bei Tabellen sowohl Überschriften (`<caption>`) als auch das spezielle Tabellenkopf-Tag `<th>`.
  Fügen Sie in das `<table>`-Tag eine Kurzbeschreibung der Tabelle mittels `summary`-Attribut ein.
  Vermeiden Sie Tabellen mit festen Breitenangaben: entweder keine Angaben oder nur %-Angaben.
  Benutzen Sie für Tabellenzellen die Attribute `scope` und `axis`.

- Sie können die Bedeutung von Akronymen und Abkürzungen über die speziellen Tags `<abbr>` und `<acronym>` einfach zugänglich machen.

Momentan wird diese Möglichkeit nur von Mozilla unterstützt

- Bitte weisen Sie mit dem `lang`-Attribut aus, wenn sich die Sprache in Ihrem Dokument zeitweilig ändert: `<span lang="en-us">This is an English text.</span>`.

- Wenn Sie Skripten mittels `<script>`-Tag in Ihr Dokument einbauen, so versehen Sie sie mit einer alternativen Textausgabe mittels `<noscript>`-Tag, um im Falle der Nichtausführbarkeit noch eine Reaktion zu ermöglichen.

- Wenn Sie die Abfrage von Mausereignissen integrieren, achten Sie bitte darauf, dass auch vergleichbare Tastaturereignisse behandelt werden:

Maus	Tastaur
onmousedown onmouseup onmouseclick onmouseover onmpuseout	onkeydown onkeyup onkeypress onfocus onblur

Für die Mausereignisse ondblclick und onmousemove gibt es keine Tastaturentsprechungen.

- Achten Sie auf hohen Kontrast und gute Lesbarkeit.

- Versehen Sie alle Seiten mit einer einheitlichen, immer an derselben Stelle angebrachten Navigationsleiste.

- Vermeiden Sie animierte GIF-Dateien. Wenn sie gebraucht werden, so achten Sie darauf, dass der Bildwechsel nur langsam erfolgt.

- Suchmöglichkeiten und Nutzerkonfigurationsmöglichkeiten könnten Ihre Site vervollständigen.

9. Anhang

9.1. Ersatz von Tags zum Zeilen- bzw. Wortumbruch

Nachfolgende Tags sind proprietär, sie werden im HTML-4.0-Standard nicht mehr unterstützt.

```
<nobr>Text Text</nobr>
```

- Kennzeichnet einen Text, der nicht umgebrochen werden darf.

<nobr> ist ersetzbar durch nichtumbrechbaren Leeraum   ( ).

```
<wbr>
```

- Tag für möglichen Umbruch im Wort (word break).
- Tag gehört nicht zum HTML-Standard und funktioniert unter Umständen nicht sauber (das Wort wird evtl. über einen Tabellenzellenrand geschrieben).
- Im XHTML wäre es als <wbr /> anzugeben (aber nicht definiert).

<wbr> ist ersetzbar durch Systemtrennstrich (soft hyphen) ().

9.2. Ersatz des <embed>-Tags

```
<embed src="URL">
```

Hiermit erfolgt die Einbettung einer Datei in einem Fremdformat über ein entsprechendes Plugin. Der Browser kann das Fremdprogramm mit der betreffenden Datei aber nur starten, wenn die Verknüpfung zwischen der Dateiendung und dem Fremdprogramm bekannt ist. Der Aufruf über das `<object>`-Tag sollte dem `<embed>`-Tag vorgezogen werden.

Das `<embed>`-Tag ist proprietär und nicht Bestandteil der HTML 4.01-Spezifikation. An seiner Stelle sollte das `<object>`-Tag eingesetzt werden. Auch wird dieses Tag von einigen Browsern nicht unterstützt.

9.3. Zum <font>-Tag

Das Font-Tag sollte eogentlich nicht mehr zum Einsatz kommen; es ist aber im HTML-4.0- und im XHTML-1.0-Standard (Transitional) noch enthalten.

```
<font size="[+|-]nummer" color="farbe" face="font-family">...</font>
```

Optionale Attribute:

- `size="[+|-]nummer"`: gibt die Schriftgröße als Zahl absolut (1 bis 7; Normalgröße ist 3) oder relativ mit Vorzeichen an.
- `face="font-family"`: Angabe der Schriftart.
- `color="farbe"`: Angabe der Schriftfarbe in hexadezimaler Form (#rrggbb) oder als Farbwort.

Eins der Attribute muss natürlich angegeben werden. `<font>` darf nur Inline-, aber keine Blockelemente enthalten, es ist daher z.B. innerhalb des `<p>`-Tags zu formulieren.

Die Zuordnung der Größenabgaben 1 bis 7 ist nicht einheitlich unter den Browsern (kursiv: Werte für font-size in Stilvorlagen). Als Orientierung seinen nachfolgende Werte angegeben: `size="1"` entspricht 8pt (0,67em; *xx-small*), `size="2"` 10pt (0,83em; *x-small*), `size="3"` 12pt (1,0em; *small*), `size="4"` 14pt (1,17em; *medium*), `size="5"` 18pt (1,5em; *large*), `size="6"` 24pt (2,0em; *x-large*) und `size="7"` 36pt (3,0em; *xx-large*).

10. Index