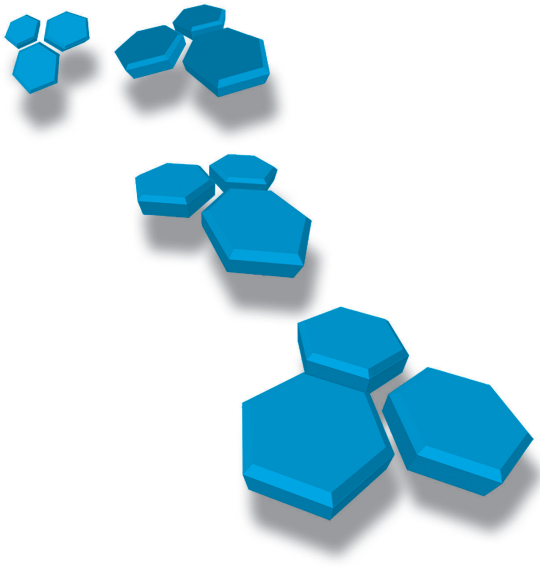
PEARSON Education

ADDISON-WESLEY

Marianne Hauser, Tobias Hauser, Christian Wenz

Das HTML/CSS Codebook



ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

3 HTML-Grundgerüst

In diesem Kapitel finden Sie grundlegende Informationen. Im Vordergrund stehen der Aufbau der HTML-Seite und wichtige allgemeine Fakten, die Sie rezeptübergreifend benötigen. Auch für Fortgeschrittene interessant ist die Auflistung und Beschreibung der verschiedenen DocTypes und eventuell zusätzlich die Erläuterungen zum Zeichensatz für eine HTML-Seite.

Exkurs

Begriffswirren rund um die Webs(e)ite

Webseite, HTML-Seite, Homepage, Website – vier Begrifflichkeiten, die oft beliebig durcheinander geschmissen werden. Mittlerweile hat sich allerdings eine Definition durchgesetzt:

- Eine Webseite ist eine einzelne Seite aus einem Webangebot.
- Eine HTML-Seite ist eine Seite mit dem HTML-Grundgerüst. Normalerweise ist eine HTML-Seite auch eine Webseite. Einzige Ausnahme ist der Einsatz von Frames. Frames unterteilen eine Webseite in mehrere HTML-Seiten (siehe dazu die Erläuterungen im Abschnitt »Frames«).
- Die Homepage ist die Einstiegsseite für ein Webangebot. Oftmals wird der Begriff mit dem kompletten Webangebot gleichgesetzt – vor allem bei privaten und kleinen Webangeboten. Meist ergibt sich aus dem Kontext, ob die Einstiegsseite oder das ganze Webangebot gemeint ist.
- Die Website ist ein Synonym für ein Webangebot. *http://www.spiegel.de/* wäre die Website des SPIEGEL (mit allen darunter liegenden Webseiten), *http://www.spiegel.de/index.html* wäre die Homepage.

Hinweis

Die CSS-Grundlagen finden Sie im nächsten Kapitel. Dort ist erklärt, wo CSS-Befehle stehen können und wie sie angewendet werden.

3.1 Aufbau

HTML besteht wie bereits erwähnt aus Tags[37]. Beispielsweise bezeichnet der Tag `<p>` einen Absatz. Hier lernen Sie nun die Umgebung kennen, in der diese Tags eingesetzt werden.

Das Grundgerüst einer HTML-Seite besteht ebenfalls aus Tags:

- Ganz außen steht das `<html>`-Tag. Dieses Tag ist das Wurzelelement der HTML-Seite. Wurzelelement heißt, es ist allen anderen Tags übergeordnet und darf nur einmal vorkommen.

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

37. Synonym zu Tag werden auch Element und Markup (von Markup Language = Beschreibungssprache) verwendet. Ein Tag ist für den Browser ein Befehl. Dennoch wird die Begrifflichkeit HTML-Befehl nicht verwendet, da sie HTML zu nah an eine Programmiersprache rücken würde. Ein Tag ist kein Programmierbefehl »Tu dies«, sondern eine Auszeichnung »Dies ist«.

- Unter dem <html>-Tag folgt als Erstes das <head>-Tag. Dies ist der Kopf der Seite (head engl. für Kopf). Dort finden Sie z.B. den Titel des Dokuments (<title>-Tag). Der Titel wird in der Titelleiste des Browsers angezeigt, ist aber auch für viele Suchmaschinen wichtig. Allgemein enthält der <head>-Bereich alles, was nicht direkt in der Seite sichtbar ist, aber für die gesamte Seite Gültigkeit besitzt.
- Das zweite Tag unter <html> ist das <body>-Tag. Hierbei handelt es sich um den Körper der Seite (body engl. für Körper). Hier finden Sie den gesamten angezeigten Inhalt, z.B. einen Absatz (<p>-Tag), Tabellen, Formulare, Überschriften, Flash-Filme und vieles mehr. Allgemein gilt, dass der <body>-Tag alles enthält, was im Inhaltsfenster des Browsers auftaucht.

Hier die komplette Grundstruktur mit einem Titel und einem Absatz[38]:

```
<html>
<head>
  <title>Dokumenttitel</title>
</head>
<body>
  <p>Inhalt</p>
</body>
</html>
```

Listing 3.1: Das HTML-Grundgerüst (grundgeruest.html)

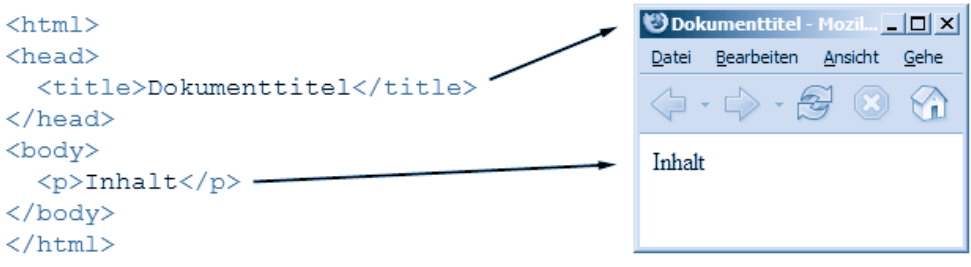


Abbildung 3.1: Der Titel landet in der Titelleiste, der Inhalt des <body>-Tag im Inhaltsfenster

Achtung

Browser sind aus historischen Gründen (siehe Kapitel 2) sehr fehlertolerant. Sie können also auch das komplette Grundgerüst weglassen und der Browser versucht immer noch, die Seite korrekt darzustellen. Meist schafft er das sogar.

Der Aufbau einer HTML-Seite ist ausgehend vom Wurzelelement <html> eine Hierarchie. Versuchen Sie sich von dieser Hierarchie eine Vorstellung zu machen. Dies hilft Ihnen vor allem beim Schreiben von Skripts mit JavaScript, aber auch bei der Verschachtelung von HTML-Elementen und der Zuweisung von CSS-Stilen.

38. Hier fehlt noch der DocType. Er wird im nächsten Abschnitt nachgereicht.

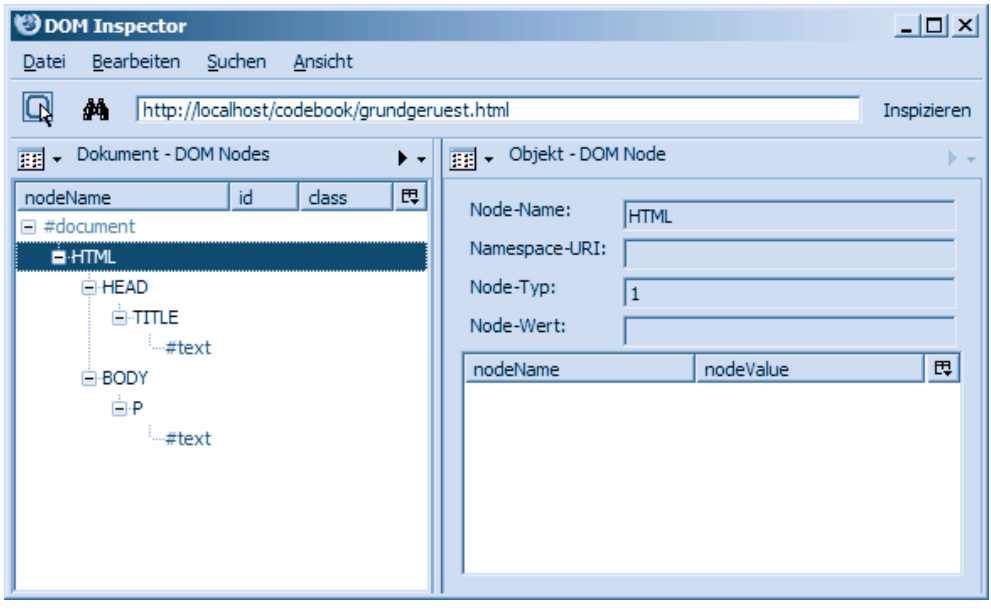


Abbildung 3.2: Wenn Sie ihn mitinstalliert haben, zeigt der DOM Inspector des Mozilla Firefox den hierarchischen Aufbau der Webseite

3.1.1 DocType

Im Grundgerüst fehlt noch der DocType als Angabe für die verwendete HTML/XHTML-Version. Den DocType benötigen Sie natürlich vor allem zum Validieren (siehe Kapitel 1). Allerdings ist dies nicht der einzige Grund. Es gibt auch einige Fälle, in denen der DocType sich direkt auf das Aussehen der Website auswirkt.

Hinweis

Einige Beispiele finden Sie in der Kategorie *Absätze*. Dort geht es darum, dass z.B. der Internet Explorer 6 die `margin`- und `padding`-CSS-Befehle nur dann spezifikationstreu umsetzen kann, wenn Sie als DocType die Strict-Variante verwenden.

Bei den DTDs in HTML- und XHTML-Dokumenten handelt es sich um den Verweis auf eine externe DTD. Die eigentliche DTD verbirgt sich unter dem dort angegebenen URL. Hier ein Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
```

Nun zu den Bestandteilen:

- `<!DOCTYPE` ist die Syntax, um auf eine DTD zu verweisen. Das Ausrufezeichen ist speziell in der DTD-Sprache wichtig und kommt sonst in HTML nicht vor.
- `html` steht für das Wurzelelement des Dokuments. In XHTML-DTDs wird es klein geschrieben, in HTML-DTDs kann es groß- oder kleingeschrieben werden, da diese nicht zwischen Groß-/Kleinschreibung unterscheiden.
- `PUBLIC` bezeichnet den Status der DTD. `PUBLIC` ist eine öffentlich verfügbare DTD. `SYSTEM` wäre eine DTD auf dem lokalen System.

- `"W3C//DTD XHTML 1.0 Transitional//EN"` ist der Identifikator der DTD-Version. Er ist entscheidend und muss immer eindeutig sein.
- `"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"` verweist auf das DTD-Dokument, in dem der komplette Standard festgehalten ist. Sie können hier auch die Kurzform verwenden:

  ```
  "DTD/xhtml1-transitional.dtd"
  ```

  oder Sie können den URL auch komplett weglassen.

HTML

Für HTML ist die Version 4.01 aktuell. Dementsprechend sollten Sie auch diese DTDs verwenden. Hier die drei möglichen DTDs für HTML 4.01:

- Transitional oder auch Loose (lose) ist die tolerante Variante der HTML-DTD. Tolerant heißt, dass Tags und Attribute die hauptsächlich zu Formatierungszwecken gebraucht werden und eigentlich von CSS-Befehlen ersetzt wurden, nach wie vor möglich sind. Solche Elemente heißen auch `deprecated` (engl. missbilligt).

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

- Strict ist wie der Name schon sagt strenger. Die nicht mehr empfohlenen Tags und Attribute dürfen mit dieser DTD auch nicht verwendet werden. Im Allgemeinen empfiehlt das W3C diese Variante.

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

- Frameset ist die Variante für den Einsatz von Frames (siehe Abschnitt »Frames«). Sie ist angelehnt an die Transitional-Variante. Der einzige Unterschied besteht darin, dass es unter dem `<html>`-Wurzelelement kein `<body>`-Element gibt, sondern stattdessen ein `<frameset>`-Element.

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

Sollten Sie die älteren Varianten benötigen, finden Sie sie in der folgenden Tabelle:

Version	DocType
HTML 4.0 Transitional	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">`
HTML 4.0 Strict	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd">`
HTML 4.0 Frameset	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN" "http://www. w3.org/TR/REC-html40/frameset.dtd">`
HTML 3.2	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">`
HTML 2.0[39]	`<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">`

Tabelle 3.1: Die DocTypes für veraltete HTML-Versionen

39. HTML 2.0 kannte außerdem noch mehrere Level. Mehr dazu unter *http://ftp.ics.uci.edu/pub/ietf/html/rfc1866.txt*.

Hinweis

Die HTML 4.0-DTD-URLs verweisen allerdings direkt auf die entsprechenden HTML 4.01-DTDs.

Hinweis

HTML-DTDs werden mit ihrem kompletten URL identifiziert. Dementsprechend können Sie natürlich auch den URL im Browser öffnen und sich die DTD ansehen.

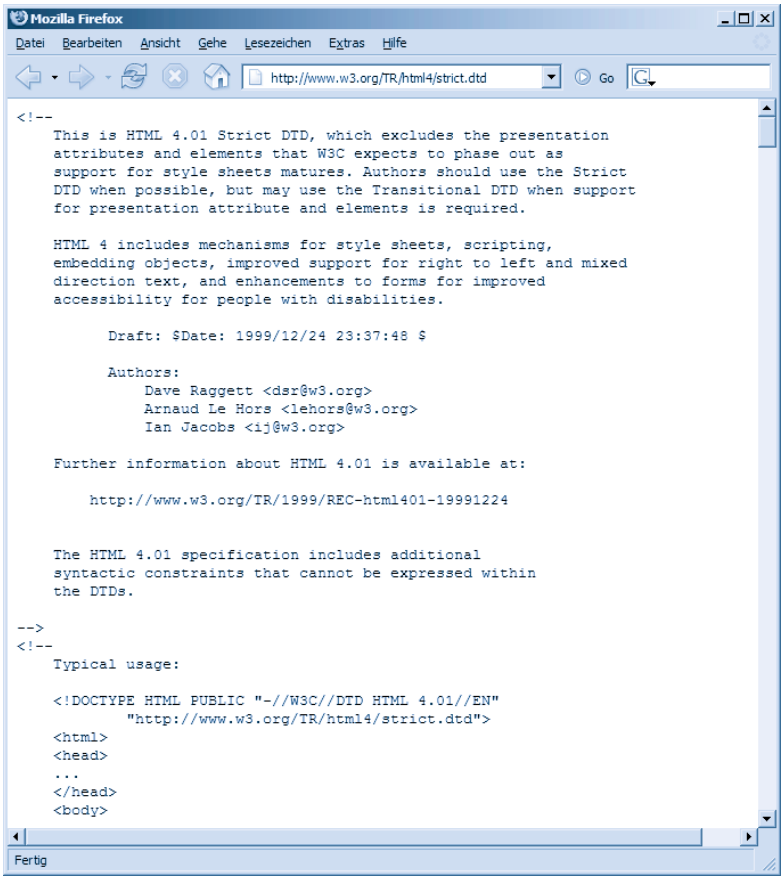


Abbildung 3.3: Der Anfang der Strict-DTD

XHTML

In der Praxis kommen heute hauptsächlich die XHTML 1.0-DTDs zum Einsatz. Auch hier gibt es drei verschiedene Versionen. Sie sind direkt an die HTML-DTDs angelehnt. Einziger Unterschied ist, dass die XHTML-DTDs schon nach den Regeln von XML[40] definiert sind.

- `Transitional` ist wie bereits erwähnt die gnädigere Variante. Im Unterschied zu HTML ist der Begriff `Loose` allerdings verschwunden.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3. org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

- `Strict` ist die strengere Alternative. XHTML 1.0 Strict ist in der Praxis das Strengste, an das man sich halten sollte. XHTML 1.1 geht mit seiner Modul-Sicht in eine andere Richtung und ist noch nicht ausreichend in den Browsern angekommen. Auch das W3C selbst orientiert seine Seiten aktuell an XHTML 1.0 Strict.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
```

- `Frameset` ist wie bei HTML von Transitional abgeleitet.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-frameset.dtd">
```

Möchten Sie die Kurzform der entsprechenden DTD, lassen Sie einfach `http://www.w3. org/TR/xhtml1/` bei dem URL weg.

Hinweis

In diesem Buch kommt im Allgemeinen die Kurzform der XHTML 1.0 Transitional-Variante zum Einsatz, da einige Rezepte auch die Formatierungstags erläutern, die als deprecated gelten. Diese »veralteten« Tags können nämlich bei besonderen Anforderungen an die Abwärtskompatibilität durchaus manchmal von Nutzen sein. Außerdem spricht dieses Buch nicht explizit von XHTML, befolgt aber sowohl den DocType als auch die Regeln. Es ist also XHTML-konform. Die Rendering-Unterschiede zwischen Transitional und Strict finden Sie in der Kategorie *Absätze*.

XHTML 1.1 und andere

XHTML 1.1 basiert direkt auf XHTML 1.0 Strict. Frames werden also z.B. nicht direkt beachtet. Dafür führt XHTML 1.1 allerdings eine modulare Sichtweise ein. Das heißt, XHTML 1.1 lässt sich mit beliebigen Modulen erweitern. Eines dieser Module können z.B. Frames sein. Die wichtigsten Änderungen von XHTML 1.0 Strict auf XHTML 1.1 finden Sie unter *http://www.w3.org/TR/xhtml11/changes.html#a_changes*.

Die Modularisierung selbst ist in der Modularization-Recommendation von 2001 festgehalten (*http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/*). Sie ist auch die Basis der weiteren auf XHTML basierenden Sprachen. So ist XHTML Basic z.B. eine leichtgewichtige Variante mit nur einigen Modulen. XHTML 1.1. und MathML 2.0 zeigt eine Kombination aus striktem XHTML mit der Formelsprache MathML.

40. Siehe dazu Kapitel 1

Version	DocType
XHTML 1.1	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">`
XHTML Basic	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN" "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">`
XHTML 1.1 und MathML 2.0	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0//EN" "http://www.w3.org/Math/DTD/mathml2/xhtml-math11-f.dtd">`

Tabelle 3.2: XHTML 1.1 und andere XHTML-DTDs

DocType-Wechsel im Browser

Browser reagieren auf den DocType vor allem im Zusammenhang mit CSS. Die Rendering-Engine des Browsers schaltet je nach DocType zwischen verschiedenen Modi um. Dieser Wechsel ist in den folgenden Browsern implementiert:

- Internet Explorer 6
- Mozilla in allen Varianten
- Opera ab Version 7
- Konqueror und Safari

Ältere Browser kennen den Wechsel nicht und interpretieren den HTML- und CSS-Code nur auf eine Art.

Standardmäßig gibt es zwei Modi, in denen Browser rendern:

- Der `QuirksMode` oder `BackCompat`-Modus ist der Kompatibilitätsmodus zu alten Browsern. Er wird unter folgenden Bedingungen verwendet:
  - wenn Sie keinen DocType einsetzen
  - wenn der DocType nicht erkannt wird, z.B. im Internet Explorer, wenn eine Processing Instruction vorangestellt wird. Letzteres ist allerdings ein Bug.
  - wenn Sie einen älteren DocType als HTML 4 einsetzen
  - wenn Sie in der DocType von HTML 4.01 Transitional keinen URL einsetzen:
    `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">`
  - wenn Sie HTML 4.0 Transitional einsetzen (egal ob mit oder ohne URL)
- Der `CSS1Compat`-Modus (auch Standard-Modus) ist der Rendering-Modus, der sich enger an die Spezifikation hält. In diesem Modus gibt es allerdings noch einmal eine Unterscheidung in den Browsern:
- Transitional-DocTypes von HTML oder XHTML werden weniger streng gehandhabt
- Strict-DocTypes und XHTML 1.1 dagegen sehr exakt

In der Praxis sollten Sie sich nicht auf den schlampigen `QuirksMode`/`BackCompat` verlassen, sondern immer eine DocType verwenden. Die Unterschiede zwischen den zwei Varianten des `CSS1Compat`-Modus werden vor allem beim CSS-Box-Modell deutlich. Dies finden Sie in der Kategorie *Absätze* in Rezept 52. Außer diesem speziellen Fall hat dieser Unterschied aber kaum Auswirkungen.

Um festzustellen, in welchem Modus eine Seite vom Browser gerendert wird, können Sie dies mit JavaScript abfangen. Der Modus wird von der Eigenschaft `compatMode` des `document`-Objekts geliefert:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
  <title>Rendering-Modus</title>
</head>
<body>
  <p>
<script language="javascript" type="text/javascript"><!--
  document.write("Rendering-Modus: " + document.compatMode);
//--></script>
  </p>
</body>
</html>
```

Listing 3.2: Den Rendering-Modus testen (test_mode.html)

Abbildung 3.4: Der Firefox im CSS1Compat-Modus

In diesem Fall erhalten Sie als Meldung `CSS1Compat`, da die Seite einen entsprechenden DocType besitzt (siehe Abbildung 3.4). In Abbildung 3.5 sehen Sie dagegen die Variante, wenn der DocType fehlt.

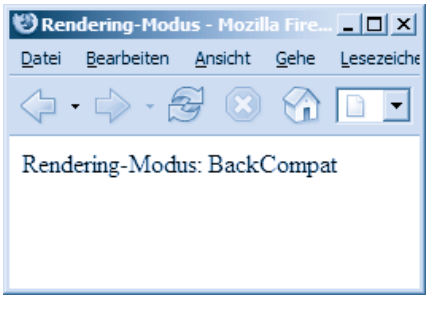


Abbildung 3.5: Ohne DocType wechselt der Browser in den BackCompat-Modus

Achtung

Die JavaScript-Eigenschaft `compatMode` kennt nicht jeder Browser. Alle Mozilla-Varianten vor der ersten Final 1 inklusive dem unsäglichen Netscape 6 liefern `undefined`. Konqueror und Safari liefern nur bei fehlendem DocType `BackCompat`, obwohl Sie auch bei älteren DocTypes in diesen Modus schalten.

Processing Instruction

Die Processing-Instruction ist das `<?xml ?>`-Tag vor jedem XML-Dokument. Sie heißt deswegen Processing-Instruction, da sie dem Parser (Prozessor) angibt, dass es sich um XML handelt. Sie kann folgende Angaben enthalten:

- Die XML-Versionsnummer. Dies ist Version 1.0. 1.1 gibt es zwar bereits, aber für XHTML ist die relevante Version 1.0.

  ```
  <?xml version="1.0"?>
  ```

- Die zweite Angabe ist der Zeichensatz. UTF-8 ist der Standard (siehe dazu den Abschnitt »Zeichensätze und Sonderzeichen«).

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  ```

Da XHTML auch ein XML-Dokument ist, ist eigentlich die Processing-Instruction vor dem Dokument notwendig. Wegen der Probleme, die sehr alte, aber auch neuere Browser mit der Processing-Instruction haben, wird sie allerdings oft weggelassen.[41]

Achtung

Speziell der Internet Explorer und der Opera 7 von den Versionen 7.00 bis 7.03 haben Schwierigkeiten mit der Processing-Instruction. Sie finden den DocType dann nicht mehr und wechseln entsprechend in den falschen CSS-Kompatibilitätsmodus.

Dies entspricht sogar einer Empfehlung des W3C (*http://www.w3.org/TR/xhtml1/#C_1*). Allerdings wird dort auch betont, dass dann die Möglichkeit entfällt, den Zeichensatz für das Dokument zu wählen. Hintergründe zu den Zeichensätzen erfahren Sie im Abschnitt »Zeichensätze und Sonderzeichen« in diesem Kapitel.

3.1.2 HTML-Tag

Das `<html>`-Tag selbst ist wie bereits erwähnt das Wurzelelement der Seite. Eigentlich ist es nicht besonders interessant. Es muss halt da sein, erfüllt aber sonst keine offensichtlich erkennbare Funktion. Dies ändert sich, wenn Sie sich die möglichen Attribute näher ansehen.

Attribute

Zunächst besitzt das `<html>`-Tag die so genannten Universalattribute oder Standardattribute. Diese drei Attribute `id`, `dir` und `lang` stehen für jedes HTML/XHTML-Tag zur Verfügung. Sie finden eine ausführliche Erläuterung im Abschnitt »Universalattribute«.

Für den Internet Explorer 6 gibt es noch das Attribut `scroll`.[42] Damit könnten Sie steuern, ob eine Scrollleiste neben oder unter dem Browserfenster auftauchen soll, wenn der Inhalt nicht komplett dargestellt werden kann. Allerdings funktioniert dieses Attribut im Internet Explorer

41. Ebenfalls problematisch kann die Processing Instruction in PHP sein, denn PHP verwendet `<? ?>` als Kurzform, um seinen eigenen Code zu markieren.
42. Zu finden unter *http://msdn.microsoft.com/workshop/author/dhtml/reference/properties/scroll.asp*

nur im CSS1Compat-Modus im Internet Explorer 6 im HTML-Tag. Im BackCompat-Modus und in älteren Internet Explorer-Versionen gehört es dagegen in das <body>-Tag (mehr dazu im Abschnitt »Attribute des <body>-Tag«).

XHTML-Besonderheiten

Für XHTML gibt es im <html>-Tag noch zwei Besonderheiten. Zum einen existiert das Universalattribut xml:lang. Mehr dazu im Abschnitt »Universalattribute«.

Zum anderen können Sie einen Namespace für XHTML angeben. Der Standard-Namespace sieht so aus:

```
<html xmlns="http://www.w3.org/1999/xhtml>
  ...
</html>
```

Ein Namespace oder auch zu Deutsch Namensraum dient dazu, in ein Dokument Ordnung zu bringen. Steht ein Namensraum so wie hier gesehen in einem Tag, gilt er für alle untergeordneten Tags, die keinen eigenen Namensraum besitzen. Diese Tags gehören also zu dem Namensraum, sind ihm quasi zugeordnet. Die Zuordnung erfolgt bei einem Namensraum immer mit dem xmlns-Attribut.

Tipp

Die offizielle W3C-Spezifikation zu Namespaces finden Sie unter *http://www.w3.org/TR/REC-xml-names/*. In XML-Dokumenten sind Namespaces deutlich wichtiger als bei XHTML. Hier können Sie den Namensraum auch weglassen, da dieser in den Browsern keine Auswirkungen hat. Der Validator des W3C (*http://validator.w3.org/*) besteht ebenfalls nicht auf den XHTML-Namensraum. Wenn Sie absolut standardkonform sein möchten, sollten Sie ihn allerdings verwenden, denn die XML-Spezifikation sieht ihn vor.

3.1.3 Head-Tag

Der Head oder Kopf einer HTML-Seite enthält alle Angaben, die nicht direkt eine Ausgabe innerhalb des Browserfensters zur Folge haben. Sie werden in den nächsten Kapiteln viele Informationen kennen lernen, die in den Head gehören. Hier nur eine kurze Übersicht über die wichtigsten:

- Der Titel des Dokuments kommt in den <title>-Tag innerhalb des Head und ist obligatorisch. Sie können nirgendwo sonst den Dokumenttitel angeben. Der Titel wird in der Titelleiste des Browsers angegeben, dient aber auch Suchmaschinen als Orientierung beziehungsweise zur Zusammenfassung des Inhalts. Daraus ergibt sich natürlich, dass Sie nach Möglichkeit immer einen passenden Titel vergeben sollten. Fehlt der mal, ersetzen die meisten Browser dies durch den URL zur angezeigten Datei.[43]

  ```
  <head>
    <title>Titel des Dokuments</title>
  </head>
  ```

- Meta-Tags sind sehr vielseitig. Sie enthalten Informationen zur Seite, helfen aber auch bei automatischen Weiterleitungen und dienen zur Arbeit mit Suchmaschinen. Sie gehören alle in den Head. Näheres zu allen Meta-Tag-Varianten finden Sie in der Kategorie *Meta-Tags und Suchmaschinen*.

43. Erlaubt ist das Fehlen laut Spezifikation nicht. Aber die Browser sind hier gewohnt tolerant. Validatoren lassen das natürlich nicht durchgehen.

- CSS Stylesheets können im Kopf der Seite abgelegt werden. Mehr dazu im nächsten Kapitel 4.
- Verlinkungen auf externe Dateien finden sich ebenfalls im Kopf der HTML-Seite. Am häufigsten werden Verlinkungen mit dem `<link>`-Tag für externe Stylesheets verwendet. Mehr dazu ebenfalls in Kapitel 4.

```
<link rel="stylesheet" type="text/css" href="style.css" />
```

- JavaScript und sonstige Skripte können ebenfalls im Kopf der HTML-Seite eingesetzt werden. Sie landen in so genannten `<script>`-Tags. Die wichtigsten Attribute sind `language` für die Art der Skriptsprache (z.B. `"javascript"`, `"jscript"` oder `"vbscript"`) und `type` für den Medientyp des Inhalts (z.B. `text/javascript`).

```
<script language="javascript" type="text/javascript">
  JavaScript-Befehle
</script>
```

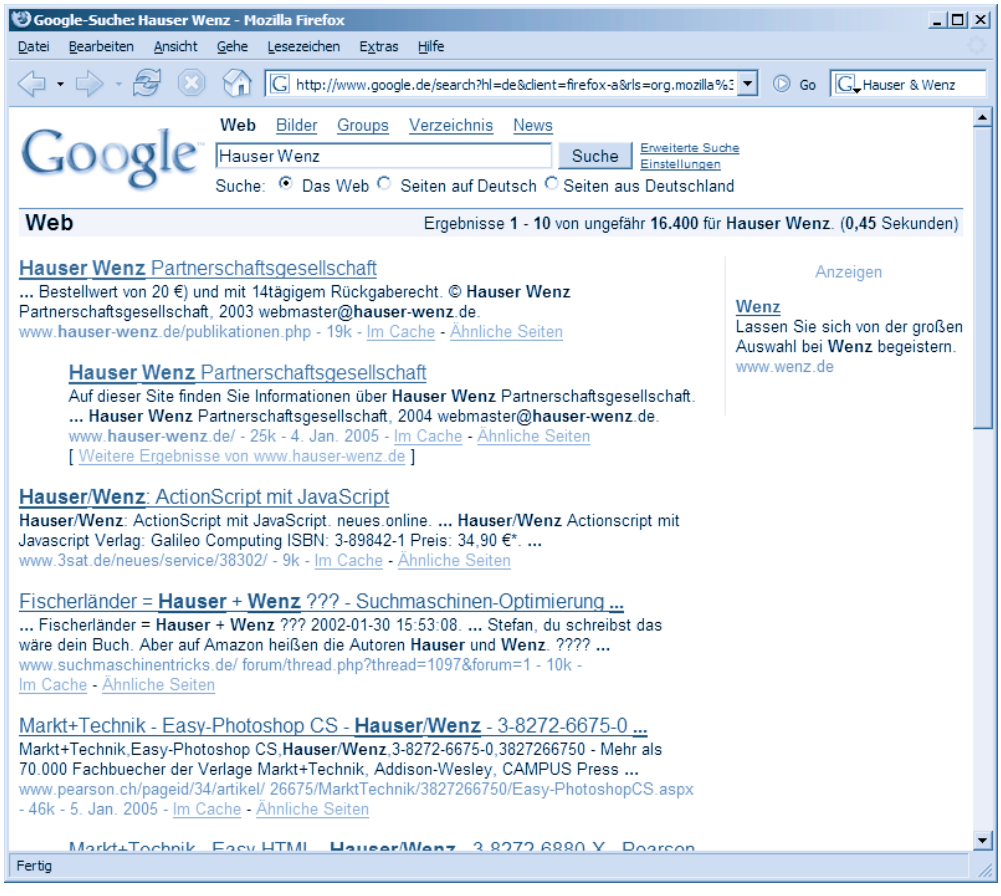


Abbildung 3.6: Der Titel unserer Website ist der Firmenname – er taucht auch bei Google auf

Achtung

Nicht jedes JavaScript-Skript kommt in den Kopf der Seite. Skripte können auch in den Body. Dies ist manchmal sogar notwendig, wenn Sie z.B. mit `document.write()` direkt einen Wert in der Seite ausgeben wollen. Würde die Anweisung im Head stehen, würde der Browser mit einem neuen, leeren Dokument beginnen.

- Weiterhin sind einige seltener gebrauchte Tags im Head möglich:
  - `<base>` legt ein Standard-Linkziel für eine Webseite fest. Siehe hierzu die Kategorie *Links und Textmarken*.
  - `<basefont>` legt eine Schriftart fest, ist aber auf den Internet Explorer und ältere Netscape-Versionen beschränkt. Siehe hierzu die Kategorie *Textgestaltung*.
  - `<bgsound>` legt ein Hintergrund-Musikstück für die Webseite fest. Auch dieses Tag ist auf den Internet Explorer beschränkt. Mehr dazu in der Kategorie *Plugins und Multimedia*.
  - `<isindex>` verweist auf einen durchsuchbaren Index für das Dokument. Ein serverseitiges Skript dazu wird mit dem `href`-Attribut verlinkt. `action` ist die Alternative für `href`. `prompt` beschreibt den Text, mit dem der Nutzer aufgefordert wird, z.B. Suchbegriffe einzugeben. Das Tag ist in HTML 4 nicht mehr empfohlen und kam auch bisher in der Praxis kaum zum Einsatz. Hier dennoch ein simples Beispiel:

```
<html>
<head>
  <title>isindex</title>
  <isindex action="search.php" prompt="Suchbegriffe eingeben " />
</head>
<body>
  <p>Inhalt der Seite</p>
</body>
</html>
```

Listing 3.3: <isindex> (isindex.html)

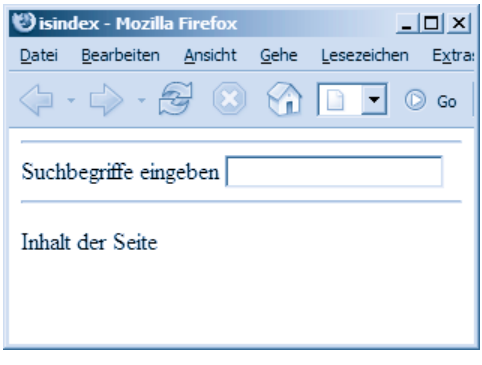


Abbildung 3.7: <isindex> setzt eine Suchmaske

Tipp

Die Browser erlauben auch den Einsatz von `<isindex>` im Körper der HTML-Seite. Dort ist er natürlich etwas sinnvoller, da Sie dann nicht vollständig an das Layout gebunden sind, das Ihnen der Browser vorgibt.

Attribute des <head>-Tag

Auch beim <head>-Tag gibt es die Universalattribute (siehe gleichnamiger Abschnitt). Dazu kommt das profile-Attribut. Mit ihm können Sie einen URL zu einer Profildatei für Metadaten angeben. Mehrere Dateien fügen Sie mit Leerzeichen aneinander. Eine Profildatei legt fest, welche <meta>-Elemente es gibt. Ein solcher Profilstandard ist z.B. von der Dublin Core Metadata Initiative (DCMI, *http://dublincore.org/*) festgelegt. Im Prinzip bräuchten Sie auch für die bekannten Attribute aus der Kategorie *Meta-Tags und Suchmaschinen* ein solches Profil, allerdings verwenden die Browser das Attribut nicht und unterschiedliche Profile machen die Auswahl problematisch.

Hinweis

In der Kategorie *Meta-Tags und Suchmaschinen* werden Sie noch eine zweite Möglichkeit finden, die Informationen der DCMI zu integrieren, nämlich über das scheme-Attribut.

3.1.4 Body-Tag

Im Körper der HTML-Seite folgen alle Inhalte, die auf der Browserseite angezeigt werden sollen. Inhalte können Überschriften, Listen, Links, Bilder, Tabellen und Absätze mit Text sein. Viele Kapitel in diesem Buch kümmern sich um diese Inhalte. Wenn Sie eine komplette Übersicht benötigen, erhalten Sie diese im Anhang.

Attribute des <body>-Tag

Neben den Universalattributen (siehe gleichnamiger Abschnitt), unterstützt das <body>-Tag einige andere. Hier eine Übersicht – jeweils garniert mit den Verweisen auf die entsprechenden Rezeptkapitel:

- Attribute für den Seitenhintergrund (siehe Kategorie *Grafiken und Hintergründe*)
- Attribute für die Seitenränder. Hier gibt es unterschiedliche Varianten (siehe Kategorie *Absätze*)
- Attribute für Linkfarben (siehe Kategorie *Links und Textmarken*)
- Attribute für JavaScript-Ereignisse (siehe Abschnitt »Ereignisse«)

Die verschiedenen Attribute zum Formatieren, also für den Hintergrund, gelten in der offiziellen Spezifikation als deprecated. Sie sollen alle durch den entsprechenden CSS-Befehl ersetzt werden. In den jeweiligen Kapiteln finden Sie immer die Tag-Lösung und im nächsten Rezept die CSS-Lösung.

Für den Internet Explorer gibt es noch das scroll-Attribut. Es funktioniert nur im BackCompat-Modus. Das Attribut akzeptiert die Werte yes (immer), auto (wenn nötig, Standardverhalten) und no (nie).

Kommentare

HTML-Kommentare können überall stehen: im Head, im Body oder auch irgendwo dazwischen. Meist werden sie in der Praxis im Body eingesetzt, um einen komplexen Seitenaufbau zu erläutern. Ein Kommentar ist Text innerhalb von <!-- und -->. Dieser Text wird vom Browser ignoriert, also nicht angezeigt. Sie sehen den Text nur, wenn Sie im Browser in den Quelltext schauen (Internet Explorer: Ansicht/Quelltext, Firefox: Ansicht/Seitenquelltext anzeigen).

Abbildung 3.8: Internet Explorer 6 ohne Scrollbalken (scroll.html)

```
<p>Sichtbarer Inhalt<!-- Inhalt des Kommentars --></p>
```

Listing 3.4: HTML-Kommentare (kommentare.html)

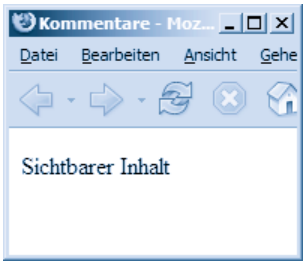


Abbildung 3.9: Nur der Text außerhalb des Kommentars ist im Browser sichtbar

Sie sollten natürlich keinen völligen Blödsinn in Kommentare schreiben, denn sie werden ja mit übertragen und vielleicht sehen Ihre Nutzer wirklich in Ihren Quellcode. In der Praxis verwenden z.B. Content Management Systeme Kommentare. Aber auch Sie selbst sollten über den Einsatz nachdenken, wenn Sie beispielsweise unterschiedliche Bereiche in der Seite kennzeichnen möchten.

Ein weiteres Einsatzgebiet ist das Auskommentieren von JavaScript-Skriptinhalt. Sehr alte Browser, z.B. der NCSA Mosaic, verstehen kein JavaScript, kennen aber auch das Tag nicht und würden sonst einfach die Skript-Befehle anzeigen. Um dies zu verhindern, legen Sie einen Kommentar darum. Da das Ende des HTML-Kommentars (`-->`) für den JavaScript-Interpreter nicht zu verstehen ist, fügen Sie davor noch einen einzeiligen JavaScript-Kommentar (`//`) ein.

```
<script language="javascript" type="text/javascript"><!--
  JavaScript-Befehle
//--></script>
```

3.1.5 Frames

Frames sind eine Erfindung aus HTML 4. Der Begriff Frame kommt vom englischen Wort für Rahmen. Hintergrund ist, dass Frames die Webseite in mehrere Rahmen aufteilen. Frames vereinen mehrere HTML-Seiten in einem Browserfenster. Jede Seite erhält einen Rahmen.

Wenn Sie für eine Seite Frames einsetzen, gibt es keinen Body. Stattdessen enthält die Seite ein so genanntes Frameset. Dieses Frameset wiederum verweist auf einzelne HTML-Seiten für die einzelnen Frames. Hier ein Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html>
<head>
  <title>Seite mit Frames</title>
</head>
<frameset cols="200, *">
  <frame name="navi" src="navigation.html" />
  <frame name="inhalt" src="inhalt.html" />
</frameset>
</html>
```

Listing 3.5: Ein einfaches Frameset (frameset.html)

Beachten Sie den geänderten DocType. Das `<frameset>`-Tag ersetzt das `<body>`-Tag. Jeder einzelne Frame wird dann wiederum durch ein `<frame>`-Tag repräsentiert. Wie die Frames verteilt sind, regelt das `cols`-Attribut. Hier finden Sie durch Kommata getrennt die Werte der Spalten. Sie können hier entweder Pixelwerte, Prozentwerte oder einen Platzhalter (*) angeben. Der Platzhalter macht den Frame so breit, dass er den übrig gebliebenen Platz im Fenster füllt.

Nun zu den einzelnen Seiten. Sie besitzen einen normalen DocType. Im Beispiel finden Sie darin nur eine Überschrift und eine Hintergrundfarbe per CSS:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
  <title>Inhalt</title>
</head>
<body style="background: rgb(255, 255, 0)">
  <h1>Inhalt</h1>
</body>
</html>
```

Listing 3.6: Die Inhalts-Seite (inhalt.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
  <title>Navigation</title>
```

Listing 3.7: Die Navigations-Seite (navigation.html)

```
</head>
<body style="background: rgb(255, 0, 0)">
  <h1>Navigation</h1>
</body>
</html>
```

Listing 3.7: Die Navigations-Seite (navigation.html) (Forts.)

Können Sie sich nun bereits vorstellen, wie das Dokument aussieht? Abbildung 3.10 verrät es. Das Dokument mit der Navigation wird in den linken Frame geladen. Dieser ist auf 200 Pixel Breite beschränkt. Der rechte Frame enthält die zweite HTML-Datei *inhalt.html*. Sie nimmt die übrige Breite des Browserfensters ein. Zwischen den Frames erscheint eine Linie.

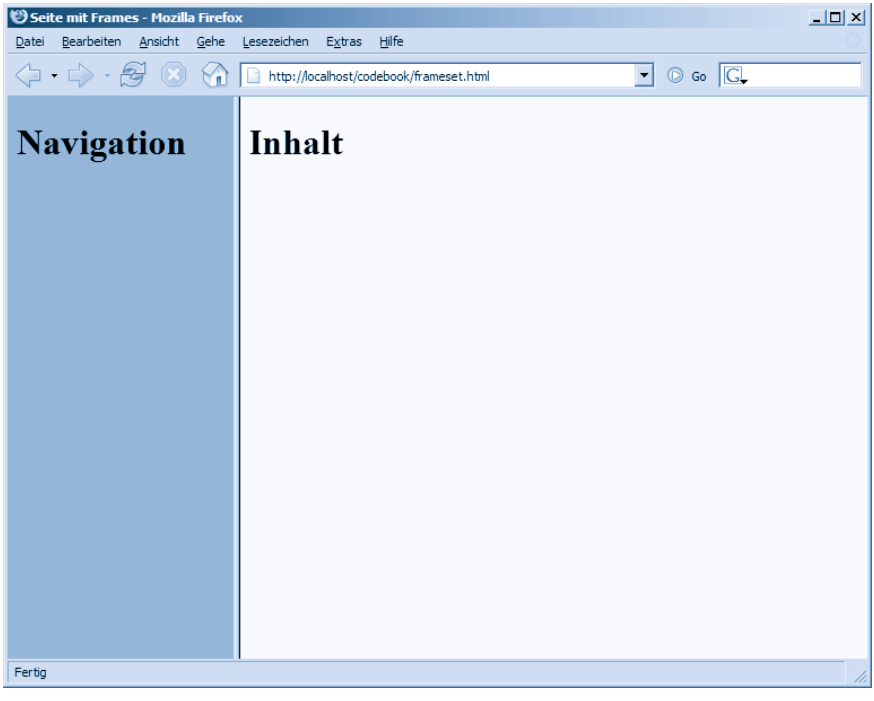


Abbildung 3.10: Zwei Rahmen: links die Navigation, rechts der Inhalt

Tipp

Diese kurze Einführung enthält natürlich nur die absoluten Grundzüge des Frame-Prinzips. In der Kategorie *Frames und Iframes* finden Sie alle Einstellungsmöglichkeiten, das Verschachteln von Frames, Links zwischen zwei oder mehr Frames und vieles mehr.

Sie können mit Frames ausgesprochen viel machen. Die Trennung von Navigation und Inhalt wird beispielsweise sehr einfach. Auch eine Fußzeile in der HTML-Seite wird einfach möglich. Allerdings sind Frames trotzdem gerade auf größeren und neueren Websites eher weniger zu finden. Dies hat mehrere Gründe:

- Da es sich um einzelne Dateien handelt, ist es schwierig, auf eine Datei aus einem Frameset ein Bookmark (Lesezeichen/Favorit – je nach Browser) zu setzen. Wird dann der Bookmark aufgerufen, ist die Navigation nicht zu sehen, wenn der Anbieter der Site nicht ein JavaScript integriert hat.

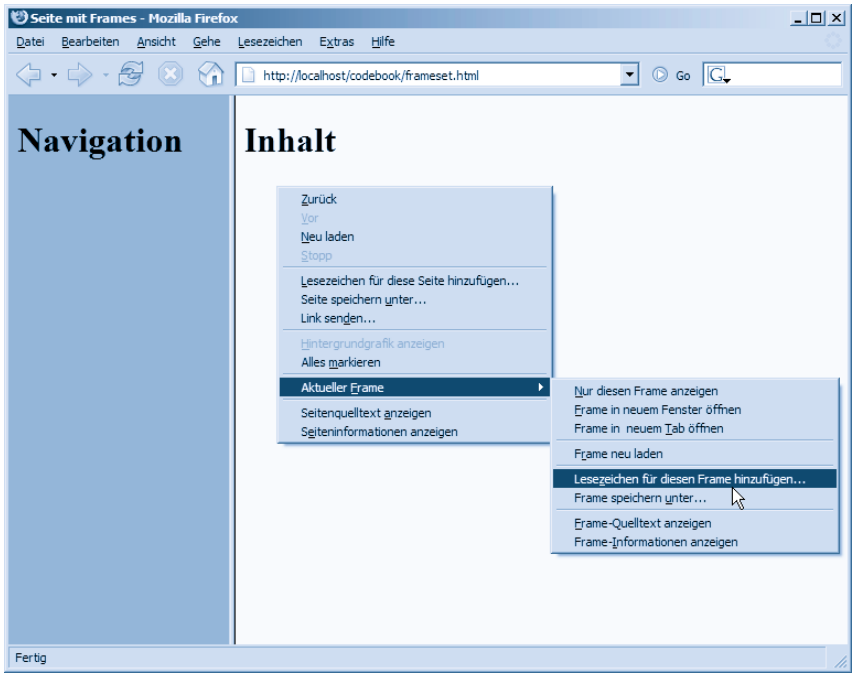


Abbildung 3.11: Möglich, aber umständlich: ein Lesezeichen für einen Frame im Firefox

- Seiten mit Frames sind für Suchmaschinen schwer indizierbar. Dies kann z.B. dazu führen, dass nur die Homepage aufgenommen wird.
- Für kleinere Projekte sind Seiten mit Frames gut wartbar. Werden es mehr Seiten und eine komplexere Navigation, wird es unübersichtlich. Gerade Content Management Systeme und serverseitige Programmierung im Allgemeinen verträgt sich nicht besonders gut mit Frames.

Zu guter Letzt ist aktuell in der Diskussion, Frames aus XHTML 2.0 komplett herauszulassen. Die Frage ist, ob Sie das schon heute davon abhalten sollte, Frames zu verwenden. Würden wir das glauben, hätten wir natürlich das Frames-Kapitel aus diesem Buch herausgelassen. Dennoch, sehen Sie sich die Alternativen an. Seitenlayout und -aufbau mit CSS ist wohl aus Sicht des W3C am zukunftsträchtigsten.

3.2 Wichtige Fakten

In diesem Abschnitt sind alle Informationen versammelt, die in mehreren Rezepten vorkommen oder die Sie für Ihre Arbeit mit den Rezepten benötigen.

3.2.1 Zeichensätze und Sonderzeichen

Um den Einsatz von Zeichensätzen und Sonderzeichen herrscht viel Verwirrung. Dies ist umso schlimmer, da man sich in Deutschland auf jeden Fall damit beschäftigen muss. Denn im Deutschen stellen die Umlaute Zeichen dar, die nicht in jedem Zeichensatz vorhanden sind. Dies können Sie sehr einfach testen. Der Firefox verwendet standardmäßig den Zeichensatz ISO 8859-1, der Umlaute enthält. Wenn Sie das umschalten und Umlaute so wie hier als normale Buchstaben im Text haben, erhalten Sie das Ergebnis aus Abbildung 3.12.

```
<p>Üäö</p>
```

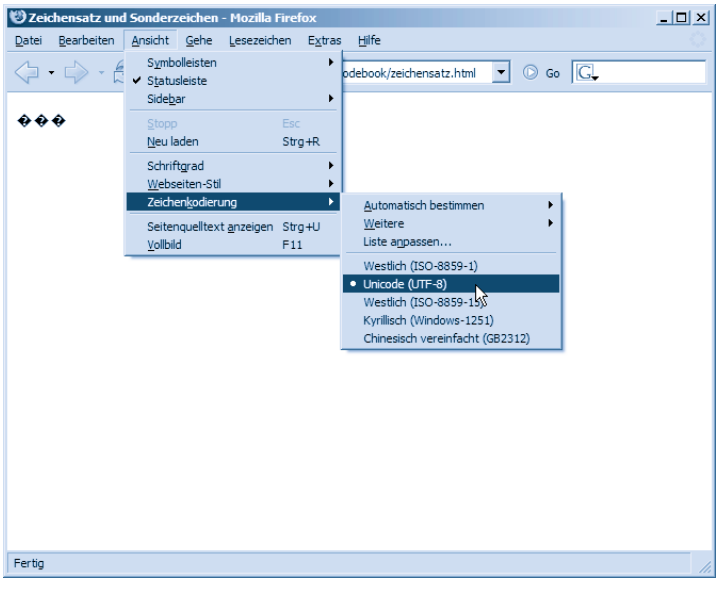


Abbildung 3.12: Der Zeichensatz lässt sich in manchen Browsern ändern. Die drei Fragezeichen repräsentieren Üäö

An sich wäre das nicht so problematisch, denn wer die Zeichenkodierung falsch einstellt, kann als selbst schuld gelten. Ein Problem sind dagegen englischsprachige Browser. Sie stellen unter Umständen Umlaute falsch dar, die einfach nur im HTML-Quelltext sind. Es gibt aber auch Situationen, bei denen zwar die Änderung der Zeichenkodierung fehlerhafte Zeichen erzeugt, bei der ein englischsprachiger Browser aber kein Problem ist. Abbildung 3.13 und Abbildung 3.14 illustrieren den Fall zweier deutscher Parteien. Die Wahl fiel auf zwei, da so das politische Gleichgewicht hergestellt ist.

Nun kennen Sie also die verschiedenen Probleme. Zeit, die verschiedenen Lösungen einzuführen.

- Die älteste besteht darin, die entsprechenden Zeichen, also z.B. die deutschen Umlaute, als Sonderzeichen darzustellen. Ein Sonderzeichen heißt auch Entität (engl. Entity). Sonderzeichen waren notwendig, da es zu Beginn des Webs keine Möglichkeit gab, den Zeichensatz festzulegen.

Abbildung 3.13: Zwei deutsche Parteien ersetzen die Umlaute nur durch komische Zeichen, wenn der Zeichensatz gewechselt wird …

Abbildung 3.14: … nicht aber bei einem englischen Browser ohne Änderung des Zeichensatzes

- Sie können einen Zeichensatz festlegen. Der Zeichensatz sollte natürlich alle Zeichen enthalten, die Sie in Ihrem Text verwenden.

In den folgenden Abschnitten finden Sie diese Alternativen kurz vorgestellt. Außerdem gehen wir den Gründen auf die Spur.

Exkurs

Zeichensätze – die Hintergründe

Computer sind per se dumme Maschinen – nicht nur der auf unserem Schreibtisch, sondern alle. Sie können also nur Zeichen darstellen, die sie kennen. Menschen müssen ihnen diese Zeichen vorgeben. Der Ausgangspunkt für jedes Zeichen ist eine Glyphe – die grafische Repräsentation eines Zeichens. Die Glyphen kommen aus den Schriftdateien. Wenn Sie also z.B. in einer Textverarbeitung einen Text eintippen, stammen die dargestellten Buchstaben aus der jeweiligen Schriftdatei. Schriften werden je nach Betriebssystem sehr unterschiedlich verwaltet. In Windows finden Sie sie z.B. im Ordner WINDOWS\FONTS. Außerdem gibt es verschiedene Formate für Schriften, z.B. TrueType, Type 1 und OpenType. Dies hat allerdings für die Arbeit des HTML- und CSS-Designers keine Bedeutung. Wichtig ist hier nur, zu wissen, dass eine Schriftart auf dem System des Nutzers verwendet wird.

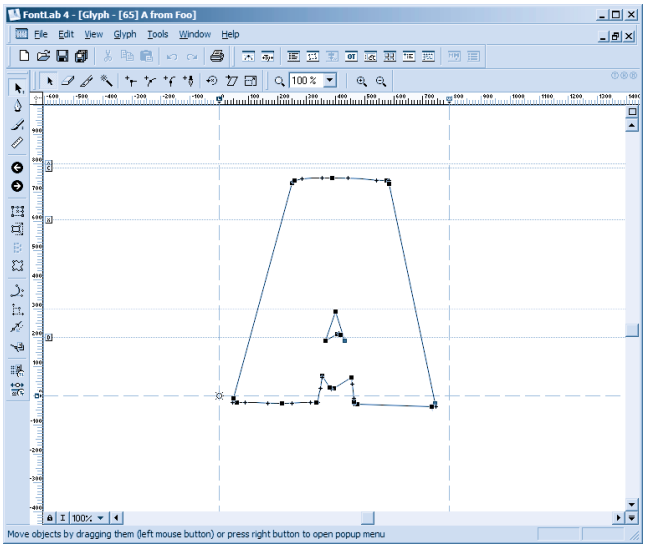


Abbildung 3.15: Eine Glyphe im Font-Programm (hier FontLab) gestalten

Glyphen allein helfen noch nichts. Glyphen müssen identifiziert werden. Sie erinnern sich, Computer sind dumm und wissen nicht von alleine, welche Glyphe z.B. ein »A« repräsentieren soll. Hierfür ist der Zeichensatz zuständig. Ein Zeichensatz besitzt jeweils einen Code für jedes Zeichen.

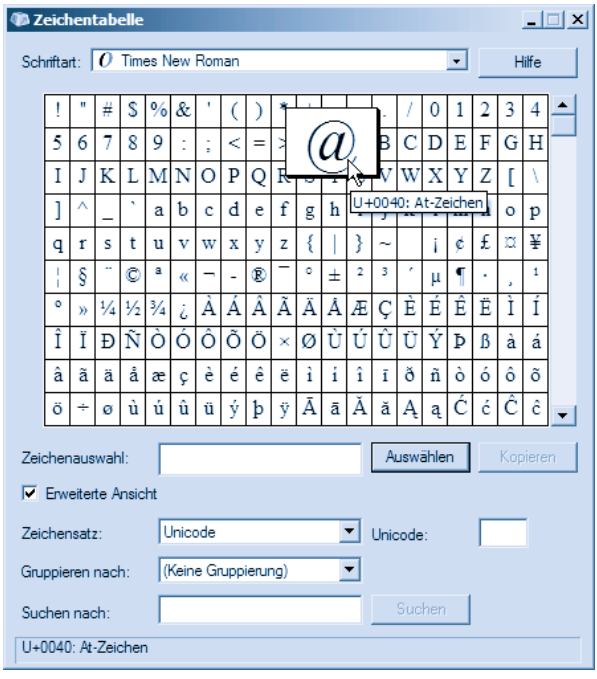


Abbildung 3.16: Die Zeichentabelle von Windows zeigt den Unicode-Zeichencode für jedes Zeichen

Der ursprüngliche Zeichensatz ist der ASCII-Zeichensatz.[44] Er besteht in seiner Urform aus 128 Zeichen (7-Bit = 2^7). Der ASCII-Code definiert jeweils eine numerische Repräsentanz für jedes Zeichen. Die sichtbaren Zeichen reichen von 32 bis 126.[45] Grundlage der Buchstaben ist das lateinische Alphabet, wie es in der Ursprache des ASCII-Codes – Englisch – eingesetzt wird. Die Großbuchstaben von 65 bis 90, die Kleinbuchstaben von 97 bis 122. Umlaute sind in den 128 Zeichen nicht repräsentiert. Sie können jedes der ASCII-Zeichen auch in HTML einsetzen. Dazu verwenden Sie die Syntax:

```
&#Zeichencode;
```

Wobei der Zeichencode für den jeweiligen dezimalen Zahlenwert steht. Tabelle 3.3 zeigt alle ASCII-Werte.

44. ASCII steht für American Standard Code for Information Interchange. Entwickelt hat sich der ASCII-Code von den für Telex (Fernschreiber) üblichen Codes. Wichtig in dieser Entwicklung waren vor allem der Baudot-Code, benannt nach dem Mathematiker Jean-Maurice-Émile Baudot, und der Murray-Code, benannt nach Donald Murray.
45. Die übrigen Zeichen sind so genannte Kontrollzeichen. Sie enthalten Steuerungsbefehle wie z.B. die ESC-Taste (ASCII-Code 27). Die meisten davon kommen aus dem ursprünglichen Einsatzgebiet von ASCII für Datenfernübertragung und dienten zur Steuerung der Druckerausgabe.

Zeichencode (dezimal)	Zeichencode (binär)	Zeichencode (hexadezimal)	Zeichencode (HTML)	Zeichen
32	0010 0000	20	&#32;	Lehrzeichen (auch SP)
33	0010 0001	21	&#33;	!
34	0010 0010	22	&#34;	”
35	0010 0011	23	&#35;	#
36	0010 0100	24	&#36;	$
37	0010 0101	25	&#37;	%
38	0010 0110	26	&#38;	&t
39	0010 0111	27	&#39;	’
40	0010 1000	28	&#40;	(
41	0010 1001	29	&#41;	)
42	0010 1010	2A	&#42;	*
43	0010 1011	2B	&#43;	+
44	0010 1100	2C	&#44;	,
45	0010 1101	2D	&#45;	-
46	0010 1110	2E	&#46;	.
47	0010 1111	2F	&#47;	/
48	0011 0000	30	&#48;	0
49	0011 0001	31	&#49;	1
50	0011 0010	32	&#50;	2
51	0011 0011	33	&#51;	3
52	0011 0100	34	&#52;	4
53	0011 0101	35	&#53;	5
54	0011 0110	36	&#54;	6
55	0011 0111	37	&#55;	7
56	0011 1000	38	&#56;	8
57	0011 1001	39	&#57;	9
58	0011 1010	3A	&#58;	:
59	0011 1011	3B	&#59;	;
60	0011 1100	3C	&#60;	<
61	0011 1101	3D	&#61;	=
62	0011 1110	3E	&#62;	>
63	0011 1111	3F	&#63;	?
64	0100 0000	40	&#64;	@
65	0100 0001	41	&#65;	A
66	0100 0010	42	&#66;	B
67	0100 0011	43	&#67;	C

Tabelle 3.3: Die sichtbaren ASCII-Zeichen

Zeichencode (dezimal)	Zeichencode (binär)	Zeichencode (hexadezimal)	Zeichencode (HTML)	Zeichen
68	0100 0100	44	&#68;	D
69	0100 0101	45	&#69;	E
70	0100 0110	46	&#70;	F
71	0100 0111	47	&#71;	G
72	0100 1000	48	&#72;	H
73	0100 1001	49	&#73;	I
74	0100 1010	4A	&#74;	J
75	0100 1011	4B	&#75;	K
76	0100 1100	4C	&#76;	L
77	0100 1101	4D	&#77;	M
78	0100 1110	4E	&#78;	N
79	0100 1111	4F	&#79;	O
80	0101 0000	50	&#80;	P
81	0101 0001	51	&#81;	Q
82	0101 0010	52	&#82;	R
83	0101 0011	53	&#83;	S
84	0101 0100	54	&#84;	T
85	0101 0101	55	&#85;	U
86	0101 0110	56	&#86;	V
87	0101 0111	57	&#87;	W
88	0101 1000	58	&#88;	X
89	0101 1001	59	&#89;	Y
90	0101 1010	5A	&#90;	Z
91	0101 1011	5B	&#91;	[
92	0101 1100	5C	&#92;	\
93	0101 1101	5D	&#93;	]
94	0101 1110	5E	&#94;	^
95	0101 1111	5F	&#95;	_
96	0110 0000	60	&#96;	`
97	0110 0001	61	&#97;	a
98	0110 0010	62	&#98;	b
99	0110 0011	63	&#99;	c
100	0110 0100	64	&#100;	d
101	0110 0101	65	&#101;	e
102	0110 0110	66	&#102;	f
103	0110 0111	67	&#103;	g
104	0110 1000	68	&#104;	h

Tabelle 3.3: Die sichtbaren ASCII-Zeichen (Forts.)

Zeichencode (dezimal)	Zeichencode (binär)	Zeichencode (hexadezimal)	Zeichencode (HTML)	Zeichen
105	0110 1001	69	&#105;	i
106	0110 1010	6A	&#106;	j
107	0110 1011	6B	&#107;	k
108	0110 1100	6C	&#108;	l
109	0110 1101	6D	&#109;	m
110	0110 1110	6E	&#110;	n
111	0110 1111	6F	&#111;	o
112	0111 0000	70	&#112;	p
113	0111 0001	71	&#113;	q
114	0111 0010	72	&#114;	r
115	0111 0011	73	&#115;	s
116	0111 0100	74	&#116;	t
117	0111 0101	75	&#117;	u
118	0111 0110	76	&#118;	v
119	0111 0111	77	&#119;	w
120	0111 1000	78	&#120;	x
121	0111 1001	79	&#121;	y
122	0111 1010	7A	&#122;	z
123	0111 1011	7B	&#123;	{
124	0111 1100	7C	&#124;	|
125	0111 1101	7D	&#125;	}
126	0111 1110	7E	&#126;	~

Tabelle 3.3: Die sichtbaren ASCII-Zeichen (Forts.)

Der ASCII-Zeichensatz hat natürlich nicht ausgereicht und wurde deswegen auf 256 Zeichen (8-Bit oder 2^8) erweitert. Allerdings herrschte das Problem, dass verschiedene Varianten die zusätzlichen 128 Zeichen unterschiedlich belegen.

Die wichtigste Lösung für Zeichensätze ist Unicode. Herausgegeben wird Unicode vom Unicode Consortium (*http://www.unicode.org/*). Die erste Version stammt von 1991, aktuell ist die vierte (genauer 4.0.1[46], 4.1 in Beta). Unicode vereinigt alle Zeichen bisheriger Zeichensätze und versucht auch, so viele Sprachzeichen wie möglich abzudecken. Ursprünglich wurde Unicode mit 16 Bit (65.536 Zeichen) angelegt. Mittlerweile wurde dies erweitert auf potentiell 1.114.112 ($2^{20} + 2^{16}$) Zeichen, von denen aber aktuell nur 96.382 belegt sind. Jedes Zeichen besitzt im Unicode-Zeichensatz eine eindeutige Nummer. Nun muss das Ganze allerdings digitalisiert werden. Das heißt, aus einer Nummer muss eine Byte-Folge entstehen. Dafür ist UTF zuständig. UTF steht für Unicode Transformation Format. Dieses Format gibt es in drei Ausprägungen UTF-32, UTF-16 und UTF-8.

46. 4.0.1 enthält allerdings gegenüber 4.0 keine neuen Zeichen.

Die Zahl legt immer fest, wie viele Bit zur Bildung eines eindeutigen Codes für ein Zeichen zur Verfügung stehen. Bei UTF-32 haben Sie also 32 Bit pro Zeichen. Da aber nur die hexadezimalen Zahlenbereiche von 0 bis 10FFFF von UTF-16 belegt sind, wird auf die nachfolgenden Stellen verzichtet. Deswegen belegt UTF-32 auch für die meisten Anwendungen gegenüber UTF-16 zu viel Platz. UTF-8 ist aus dem Wunsch entstanden, eine schlanke Lösung zu haben, die alle Zeichen abdeckt und platzsparender ist. Bei UTF-8 erhält jedes Zeichen eine Byte-Kette. Die Länge ist flexibel von 1 bis 4 Byte. Dadurch lässt sich die komplette Bandbreite an Unicode-Zeichen abbilden.

Listen der Unicode-Zeichencodes finden Sie unter *http://www.unicode.org/charts/*. Sie können jedes Unicode-Zeichen auch in HTML kodieren. Dazu gibt es zwei Varianten. Mit vorangestelltem kleinem `x` als Hexadezimalwert[47]:

```
&#xHexadezimalwert;
```

oder ohne als Dezimalwert:

```
&#Dezimalwert;
```

Die Unicode-Bemühungen wurden zeitgleich von Projekten bei der ISO begleitet. Glücklicherweise haben sich beide Standardisierungskonsortien auf einen gemeinsamen Weg geeinigt und halten bei ihren Versionen dieselben Zeichentabellen bereit. So entspricht Unicode 4 dem ISO-Standard ISO 10646. Allein die Kodierung ist unterschiedlich. Die ISO bietet allerdings auch 8-Bit Zeichensätze unter der Norm ISO/IEC 8859. Im Gegensatz zu UTF-8 gibt es hier mehrere Varianten, die als erste 128 Zeichen die ASCII-Codes haben. Dann folgen Steuerzeichen und erst die letzten 96 Zeichen sind jeweils für die einzelnen Varianten spezifisch. Tabelle 3.4 gibt einen Überblick über die Varianten. Für deutsche Bedürfnisse geeignet sind ISO 8859-1 und ISO 8859-15. Allerdings werden beide Varianten nicht weiterentwickelt und auf Dauer geht der Weg hin zu UTF-8.

Norm	Beschreibung
ISO 8859-1	Latin-1 – Westeuropäisch
ISO 8859-2	Latin-2 – Osteuropäisch
ISO 8859-3	Latin-3 – Südeuropäisch und Esperanto
ISO 8859-4	Latin-4 – Baltisch
ISO 8859-5	Kyrillisch
ISO 8859-6	Arabisch
ISO 8859-7	Griechisch
ISO 8859-8	Hebräisch
ISO 8859-9	Latin-5 – Türkisch statt Isländisch, sonst wie Latin-1
ISO 8859-10	Latin-6 – Nordisch
ISO 8859-11	Thai
ISO 8859-13	Latin-7 – Baltisch ersetzt Latin-4 und -6
ISO 8859-14	Latin-8 – Keltisch

Tabelle 3.4: Die verschiedenen ISO 8859-Varianten

47. Die Beschreibung und Umrechnung von hexadezimalen in dezimale Werte und umgekehrt lernen Sie im nächsten Kapitel kennen.

Norm	Beschreibung
ISO 8859-15	Latin-9 – Westeuropäisch mit Eurozeichen
ISO 8859-16	Latin-10 – Südosteuropäisch mit Eurozeichen

Tabelle 3.4: Die verschiedenen ISO 8859-Varianten (Forts.)

Ein wenig Vorsicht müssen Sie walten lassen, wenn Sie Zeichensätze im anderen Gewand wieder treffen. Z.B. gibt es unter Windows den Zeichensatz Windows-1252. Er entspricht allerdings weitgehend Latin-1 (*http://www.microsoft.com/globaldev/reference/sbcs/1252.htm*). Besser ist es hier natürlich die nicht proprietäre ISO-Variante zu verwenden.

Entitäten

Eine Entität beginnt immer mit einem kaufmännischen Und (`&`).[48] Dann folgt der Name der Entität, z.B. `euro` für das Euro-Währungssymbol. Abgeschlossen wird mit einem Strichpunkt (`;`). Die Entität für den Euro sieht also so aus:

```
<p>230 &euro;</p>
```

liefert demnach die Ausgabe `230 €`.

Der Vorgang, Zeichen mit Entitäten darzustellen, heißt übrigens auch Maskieren. Nicht jede Entität hat einen Namen. Aber alle lassen sich mit ihrem Code identifizieren. Der Code basiert auf dem Unicode. Sie können also auch für das Euro-Symbol seinen Code angeben:

```
<p>230 &#8364;</p>
```

Die dritte Alternative ist die hexadezimale Darstellung. Hierfür kommt vor dem Code ein `x`. Der Code selbst ist dann in hexadezimaler Schreibweise.[49] Für den Euro sähe das so aus:

```
<p>230 &#x20AC;</p>
```

Die Entitäten für Umlaute und ein paar andere wichtige Sonderzeichen finden Sie in Tabelle 3.5, eine komplette Liste ist im Anhang zu finden.

Entität	Unicode (dezimal)	Zeichen	Beschreibung
`&Auml;`	`&#196;`	Ä	
`&auml;`	`&#228;`	ä	
`&Ouml;`	`&#214;`	Ö	
`&ouml;`	`&#246;`	ö	
`&Uuml;`	`&#220;`	Ü	
`&uuml;`	`&#252;`	ü	
`$szlig;`	`&#223;`	ß	scharfes s
`&copy;`	`&#169;`	©	Copyright-Zeichen
	`&#39;`	’	Apostroph
`&quot;`	`&#34;`	”	Gerade Anführungsstriche

Tabelle 3.5: Wichtige Entitäten

48. Sollten Sie selbst ein kaufmännisches Und im Text verwenden wollen, müssen Sie die zugehörige Entität `&` verwenden.
49. Die Umrechnung zwischen dezimalen und hexadezimalen Werten finden Sie im nächsten Kapitel erläutert.

Entität	Unicode (dezimal)	Zeichen	Beschreibung
&euro;	&#8364;	€	Euro-Währungssymbol (nicht in sehr alten Browsern)
&lt;	&#60;	<	Kleiner als (spitze Klammer auf)[50]
&gt;	&#62;	>	Größer als (spitze Klammer zu)
&amp;	&#38;	&	Kaufmännisches UND

Tabelle 3.5: Wichtige Entitäten (Forts.)

Zeichensatz festlegen

Wenn Sie nicht alle deutschen Umlaute abtippen möchten, können Sie auch einen Zeichensatz für das Dokument festlegen. Dazu gibt es prinzipiell zwei Orte:

- In der Processing-Instruction.

```
<?xml encoding="ISO-8859-15"?>
```

- oder in einem speziellen <meta>-Tag (siehe hierzu auch Kategorie *Meta-Tags und Suchmaschinen*):

```
<meta http-equiv="Content-type" content="text/html; charset=iso-8859-15" />
```

Prinzipiell ist die Variante mit dem <meta>-Tag vorzuziehen, da die Processing-Instruction auf Grund der Unverträglichkeit im Internet Explorer weniger zu empfehlen ist. Bleibt noch die Wahl des Zeichensatzes. Hier hat sich als Standard in deutschen Landen ISO-8859-1 und mit Euro-Zeichen ISO-8859-15 durchgesetzt. Allerdings werden diese Varianten von der ISO offiziell nicht weiterentwickelt.

Die Alternative ist der Einsatz von UTF-8. Der Haken hierbei ist, dass der Editor, den Sie verwenden, mit UTF-8 zurechtkommen sollte. Ein auf ASCII-basierender Texteditor kann oft kein UTF-8-Dokument speichern. In Abbildung 3.17 sehen Sie z.B. ein UTF-8-basiertes HTML-Dokument, geöffnet in einem Editor mit ASCII-Basis.

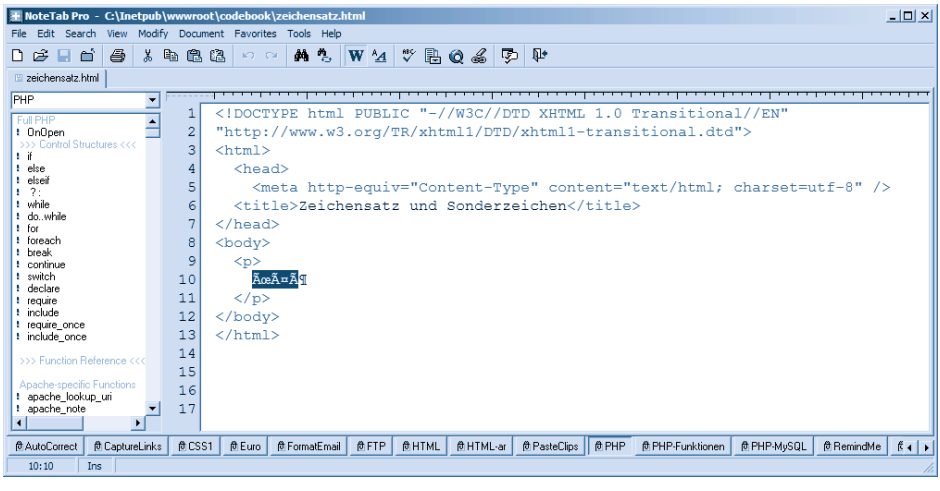


Abbildung 3.17: UTF-8-Kodierung in einem ASCII-basierten Texteditor

50. Größer als und Kleiner als wären an sich keine Sonderzeichen, da sich ihr Code innerhalb der 128 ASCII-Zeichen befindet. Sie sind allerdings in HTML spezifisch für Tags eingesetzt und werden deswegen extra als Sonderzeichen benötigt.

Nun wissen Sie auch, was das Problem in Abbildung 3.13 war. Der deutsche Browser war auf UTF-8 gewechselt. Die Seite war aber als ASCII-Dokument mit ISO-8859-1 gespeichert.

Welche Strategie?

Eine Angabe zum Zeichentyp ist auf jeden Fall kein Luxus, sondern heute schon Notwendigkeit. Der Validator des W3C mahnt sie mit Vehemenz an. Ganz abgesehen davon bedeutet dies kaum Mehraufwand.

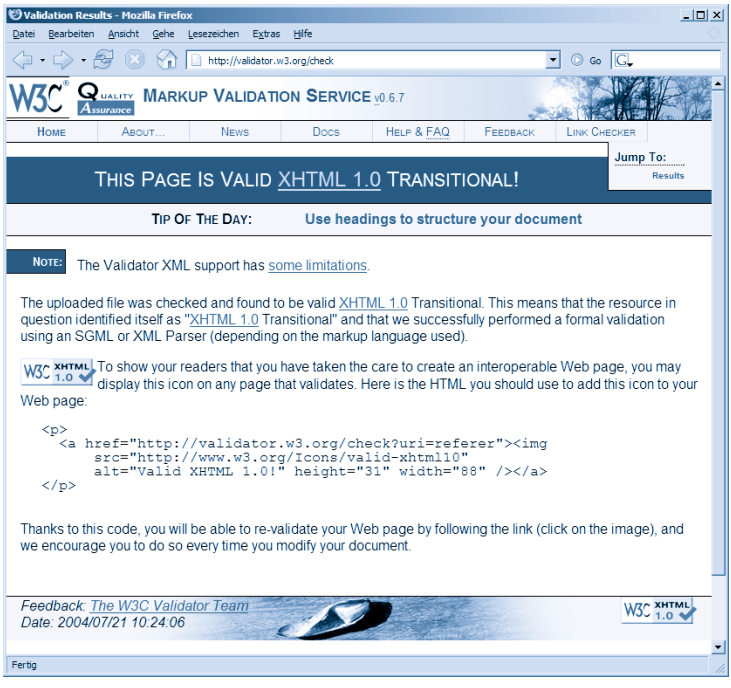


Abbildung 3.18: Nur mit Zeichensatz erhalten Sie die höchsten Weihen des W3C-Validators

Die nächste Frage ist, ob UTF-8 oder ISO-8859-1/ISO-8859-15. Prinzipiell sind beide Varianten in Ordnung. Wenn Ihr HTML-Editor mit UTF-8 zurechtkommt, können Sie dies vorziehen. Allerdings ist es durchaus auch mal ärgerlich, wenn unterwegs für die eigene Website nur ein ASCII-basierter Editor verfügbar ist. Vorsicht beim Wechseln der Varianten, viele Editoren ändern nicht sofort das Speicherformat, wenn Sie den Zeichensatz im `<meta>`-Tag ändern. Eventuell müssen Sie neu speichern.

Tipp

Wenn Sie mehr Zeichen zur Verfügung haben müssen oder global einen Zeichensatz einsetzen möchten, ist auf jeden Fall UTF-8 sinnvoll, da dieser Zeichensatz alle Unicode-Zeichen abbilden kann.

Die letzte Frage ist, ob Sie Umlaute als Entitäten darstellen sollten oder nicht. Ein paar Argumente helfen bei der Entscheidung: Sind Umlaute und Sonderzeichen als Entitäten angegeben, so ist Ihre Website sogar vor dem (evtl. unbeabsichtigten) Wechsel der Zeichenkodierung sicher.[51] Außerdem

51. Deswegen konnte die Website der CDU in (Abbildung 3.13) nicht herhalten. Ihre Umlaute sind komplett als Entitäten maskiert.

sind Sie damit auch mit uralten oder exotischeren Browsern kompatibel. Die moderneren, auch fremdsprachigen kommen mit den Zeichensätzen zurecht. Der einzige, aber durchaus gravierende Nachteil am Verwenden von Entitäten liegt im umständlichen Maskiervorgang. In diesem Buch sind die Umlaute maskiert. Die Angaben zum Zeichensatz lassen wir aus Platzgründen weg.

Tipp

Lassen Sie die Umwandlung in Sonderzeichen doch einfach Ihren Editor erledigen. Sowohl viele WYSIWYG als auch Texteditoren bieten diese Funktionalität mittlerweile von Haus aus. Auch in serverseitigen Technologien wie z.B. PHP finden Sie Automatismen. Eine sehr bekannte Bibliothek, die das erledigt, ist HTML Tidy.

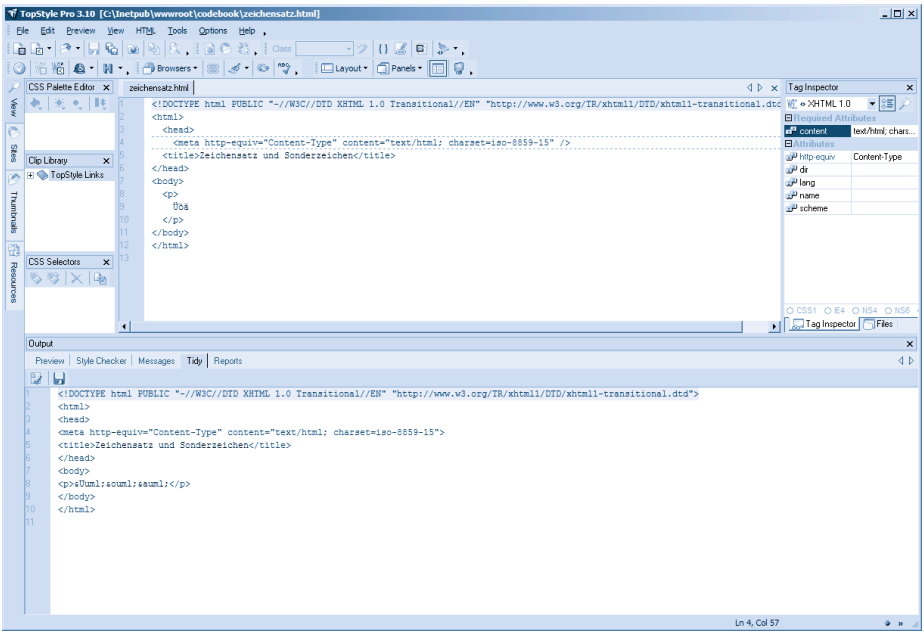


Abbildung 3.19: Oben die Ausgangssituation, unten das Ergebnis dank HTML Tidy im Editor TopStyle

3.2.2 Medientypen

Der gezeigte `<meta>`-Tag für den Zeichensatz enthält noch eine zweite Angabe, nämlich den Medientyp für die Seite. Der Standard-Medientyp für HTML ist `text/html`. Allerdings – und hier gehen die Probleme los – gilt das nur für HTML (inkl. 4.0.1), nicht aber für XHTML

Aber der Reihe nach. Normalerweise steht der Medientyp im HTTP-Header. Das ist das bereits in Kapitel 1 erwähnte HTTP-Feld `Content-type`. Hierbei erfolgt die Steuerung vom Webserver. Der Medientyp ist – wie alle im Web verwendeten Medien-/Datentypen – ein MIME-Typ. MIME steht für Multipurpose E-Mail Extension. Dieser Standard ist älter als das Web und regelt Bezeichnungen und Kategorien für verschiedene Medientypen. Offiziell standardisiert ist MIME von der IETF (*http://www.ietf.org/*) in den RFC 2045 bis 2049.

Exkurs

MIME-Typen im Webserver

Die erste Anlaufstelle zum Angeben oder Ändern von MIME-Typen ist der Webserver. Im Apache, dem aktuell am weitesten verbreiteten Webserver[52], finden Sie die MIME-Typen in der Datei *mime.types* im Verzeichnis *conf*. Die dort angegebenen MIME-Typen werden von Apache für den HTTP-Header verwendet.

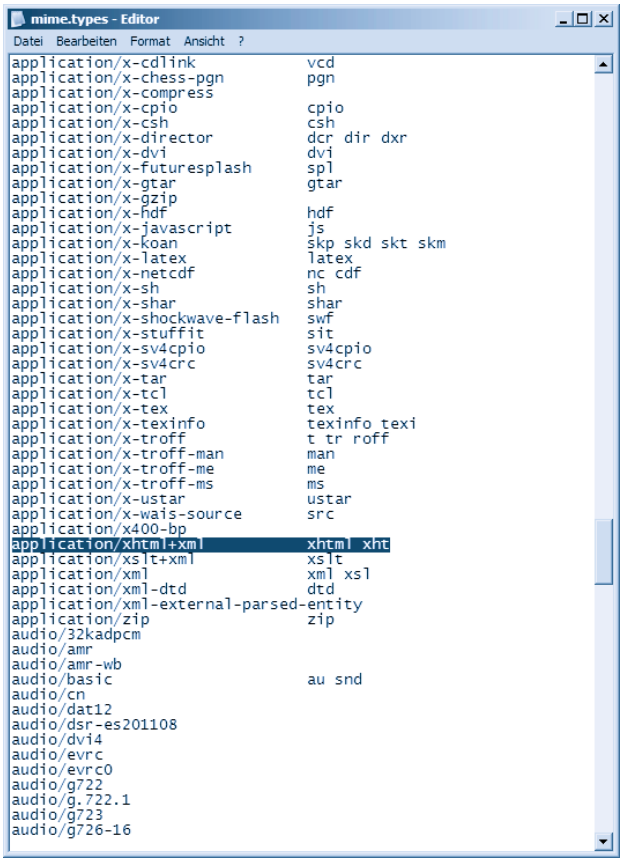


Abbildung 3.20: Die Datei mime.types enthält bereits eine Definition für .xhtml

52. Eine Bemerkung mit durchaus politischer Brisanz. Immerhin tobt aktuell ein Kampf der Statistikinstitute vor allem zwischen Netcraft (*http://www.netcraft.com/*) und Port80, einer kleinen Firma, die die Webserver der größten 1000 Unternehmen nach Fortune in der Studie aufnimmt (*http://www.port80software.com/*).

Zum Hinzufügen im Apache verwenden Sie besser den Befehl `AddType` in der Konfigurationsdatei *httpd.conf* (im Ordner *conf*), als direkt *mime.types* zu editieren. Dies sieht z.B. so aus:

```
AddType application/xhtml+xml .xhtml
```

Diesen MIME-Type müssen Sie allerdings nicht anlegen, da er bereits vorhanden ist.

Im IIS (Internet Information Service) fügen Sie neue MIME-Typen so hinzu: Wechseln Sie zuerst in die SYSTEMSTEUERUNG und dort in VERWALTUNG. Öffnen Sie dann über INTERNET-INFORMATIONSDIENSTE die Managementkonsole des IIS und klappen Sie die Website aus, für die Sie den MIME-Type setzen möchten. Sie finden die nötigen Einstellungen dann, wenn Sie mit der rechten Maustaste auf die Website klicken und EIGENSCHAFTEN wählen. Im Register HTTP-HEADER finden Sie die Schaltfläche DATEITYPEN und dort können Sie mit NEU einen neuen hinzufügen.[53]

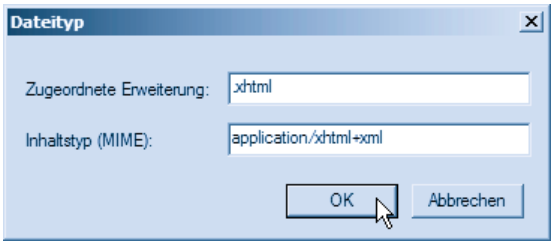


Abbildung 3.21: Im Microsoft Internet Information Server einen MIME-Type hinzufügen

Normale HTML-Dokumente haben den MIME-Type `text/html`. Für XHTML-Dokumente ist das aber eigentlich nicht vollständig richtig. Das W3C fasst die Situation in einer Note zusammen (*http://www.w3.org/TR/xhtml-media-types/*). Die Essenz daraus sehen Sie in Tabelle 3.6.

Medientyp:	text/html	application/xhtml+xml	application/xml	text/xml
HTML 4.01	Sollte	Darf nicht	Darf nicht	Darf nicht
XHTML 1.0 Transitional	Darf	Sollte	Darf	Darf
XHTML 1.0 Strict	Darf nicht	Sollte	Darf	Darf
XHTML 1.1 / Basic	Darf nicht	Sollte	Darf	Darf
XHTML 1.1 + MathML 2.0	Darf nicht	Sollte	Darf	Darf

Tabelle 3.6: MIME-Typen und die zugehörigen W3C-Empfehlungen[54]

53. Dieses Dokument erklärt das Vorgehen für IIS 6.0 (Windows 2003): *http://www.microsoft.com/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/Default.asp?url=/resources/documentation/windowsserv/2003/standard/proddocs/en-us/wsa_mimemapcfg.asp. Für IIS 5.0 (Windows 2000 Server, Windows XP) finden Sie das Vorgehen hier: http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/iis/maintain/featusability/mimeiis.mspx*

54. Angelehnt an *http://www.w3.org/TR/xhtml-media-types/#summary* mit Anpassungen. Die normativen Begriffe sind folgendermaßen übersetzt (von stark zustimmend bis stark ablehnend): Should = Sollte, May = Darf, Should not = Sollte nicht, Must not = Darf nicht

Das heißt also, dass Sie für HTML nur `text/html` einsetzen dürfen, für XHTML aber auch schon in der Transitional-Variante `application/xhtml+xml` wählen sollten. Denn Fakt ist, dass sonst zwar der DocType verwendet wird, um den Modus für CSS festzustellen, der Browser aber trotzdem nicht wirklich XHTML interpretiert. In der Praxis macht das insofern einen Unterschied, da der Browser bei XHTML die XML-Regeln so strikt auslegen müsste, dass falsche Verschachtelungen dazu führen, dass ein Dokument überhaupt nicht angezeigt wird. Außerdem müsste der Browser z.B. auch bei CSS-Zuweisungen Groß- und Kleinschreibung unterscheiden. Dass dies erhöhte Anforderungen an den Webmaster stellen würde, ist klar.

Nun aber genug im Konjunktiv geredet. Aktuell sollten Sie `text/html` vorziehen. Genauer gesagt ist ein Wechsel des MIME-Type aktuell vor XHTML 2.0 (siehe Kapitel 5) nicht sinnvoll. Dies hat mehrere Gründe:

- Nur wenige Browser unterstützen den neuen MIME-Type. Der Internet Explorer tut dies erst seit dem zweiten Servicepack von Version 6. Die beste Unterstützung bringen aktuell die Mozilla-basierenden.[55]

Tipp

Unterstützt ein Browser `application/xhtml+xml`, so liefert er diesen MIME-Type auch im `HTTP_ACCEPT`-Feld seiner HTTP-Anfrage. Diese können Sie natürlich mit einer serverseitigen Technologie wie PHP auslesen und entsprechend den Header setzen:

```
<?php
  if (stristr($_SERVER["HTTP_ACCEPT"], "application/xhtml+xml")) {
    header("Content-type: application/xhtml+xml");
  } else {
    header("Content-type: text/html");
  }
?>
```

Listing 3.8: Auf den unterstützten MIME-Typ reagieren (mime.php)

Vorsicht, das Skript muss vor allen Ausgaben per PHP oder HTML liegen, da der HTTP-Header am Anfang der Seite vor jeder Art von Inhalt gesetzt sein muss.

- Beim JavaScript-Zugriff bleibt kaum ein Stein auf dem anderen. Da das Dokument XML ist, müssen Sie über Knoten und ID navigieren. Für die Kundigen: `document.forms`, `document.images` etc. gibt es nicht, sie können also nicht wie in JavaScript gewohnt auf Formularelemente und Bilder zugreifen. Hier müssen Sie über den Tag-Namen gehen.
- In CSS ergeben sich ebenfalls ein paar Änderungen: z.B. muss eine globale Hintergrundfarbe für das Wurzelelement, also `<html>` definiert werden und nicht für `<body>`.

Insgesamt gesehen ist also der MIME-Type `text/html` heute die beste, weil kompatibelste Wahl. Ein Wechsel auf `application/xhtml+xml` ist zwar schon möglich, aber mit so vielen Problemen behaftet, dass es sich wohl erst für XHTML 2.0 lohnt. Dort wird nur noch ein XML-affiner MIME-Typ zur Verfügung stehen.

55. Detaillierte Testergebnisse unter *http://www.w3.org/People/mimasa/test/xhtml/media-types/results*.

Tipp

Sie können in HTML auch clientseitig den MIME-Type setzen. Dies geschieht im `<meta>`-Tag, das auch für den Zeichensatz zuständig ist (Infos zu `<meta>`-Tags in der Kategorie *Meta-Tags und Suchmaschinen*):

```
<meta http-equiv="Content-type" content="text/html; charset=iso-8859-15" />
```

Allerdings sollte sich die Angabe hier nicht von der des Webservers oder der per serverseitigem Skript getroffenen unterscheiden.

3.2.3 Universalattribute

Die HTML-Spezifikation definiert einige Universalattribute, die als solche für jedes Tag anwendbar sind. Wie immer gibt es aber auch bei dieser Regel Ausnahmen. Sie finden diese in Tabelle 3.7. Nun aber zuerst einmal die Universalattribute selbst:

- `id` erlaubt die Vergabe eines eindeutigen Identifikators für ein Tag. Die ID wird von CSS und von JavaScript genutzt. Hier ein einfaches JavaScript-Beispiel, bei dem die Schriftgröße eines Absatzes geändert wird, wenn die Seite geladen ist:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
  <title>Einsatz von ID</title>
<script language="javascript" type="text/javascript"><!--
  function zoom() {
    document.getElementById("absatz").style.fontSize="30pt";
  }
//--></script>
</head>
<body onload="zoom()">
  <p id="absatz">Ein Absatz mit Text.
  </p>
</body>
</html>
```

Listing 3.9: Auf ein Element per ID zugreifen (id_einsatz.html)[56]

- `dir` legt die Richtung der Seite fest, und zwar mit den Werten `ltr` (left to right = von links nach rechts) und `rtl` (right to left = von rechts nach links). Hintergrund ist die Ausrichtung für Sprachen, die in anderer Laufrichtung gelesen werden. Die Standardeinstellung ist `ltr`. Hier ein Beispiel:

56. Das Skript ist auf W3C-DOM-kompatible Browser beschränkt und funktioniert z.B. nicht mit Netscape Navigator 4.x.

Abbildung 3.22: Der Text des per ID identifizierten Absatzes wird größer dargestellt

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html dir="rtl">
  <head>
  <title>dir-Universalattribut</title>
</head>
<body>
  <p>Ein Absatz mit Inhalt</p>
</body>
</html>
```

Listing 3.10: Die Richtung mit dir ändern (dir.html)

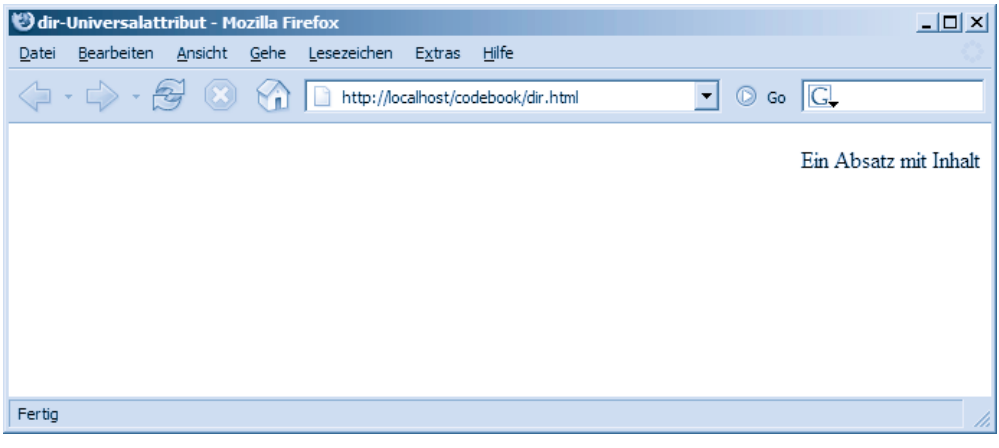


Abbildung 3.23: Der Einsatz von dir

- `lang` gibt die Sprache des HTML-Dokuments oder Elements an. Diese Angabe enthält potenziell das Sprachkürzel nach ISO 639-1. Dieser Standard definiert für viele Sprachen zweistellige Kürzel, z.B. `de` für Deutsch oder `en` für Englisch (eine Liste finden Sie im Anhang). Dreistellige Sprachkürzel befinden sich in ISO 639-2, kommen allerdings im Web selten zum Einsatz.

```
<html lang="en">
```

Zusätzlich können Länderkürzel nach ISO 3166 zum Einsatz kommen. Der Einsatz von `lang`-Attributen kann beispielsweise für Screenreader interessant sein, die wissen müssen, ob und wann sich in einer Seite die Sprache ändert (siehe hierzu die Kategorie *Accessibility und Usability*). Ebenfalls wichtig sind sie für manche Suchmaschinen. Im Text sind sie praktisch zur Wahl der Art von Anführungszeichen – allerdings unterstützt Letzteres noch kein aktueller Browser.

In XHTML wird statt des `lang`-Attributes die Variante `xml:lang` empfohlen. In XHTML 1.0 sollten beide parallel vorhanden sein, in XHTML 2.0 nur noch `xml:lang`. Hier die XHTML 1.0-Varianten:

```
<html xmlns=" http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
```

- `title` vergibt einen Namen für das jeweilige Element. Dieser Name wird sowohl für die so genannten Tooltipps, also die kleinen Hinweiskästchen im Browser, benötigt, als auch für Screenreader und ähnliche Accessibility-Tools.

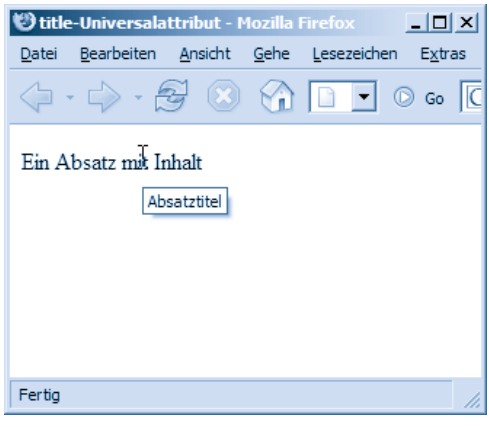


Abbildung 3.24: Der Wert des title-Attributs wird als Hinweis angegeben

- `class` und `style` dienen zum CSS-Einsatz und werden im nächsten Kapitel erläutert.

Universalattribut	Nicht vorhanden in:
class	<base>, <head>, <html>, <meta>, <param>, <script>, <style>, <title>
dir	<base>, , <frame>, <frameset>, <hr>, <iframe>, <param>, <script>
lang	<base>, , <frame>, <frameset>, <hr>, <iframe>, <param>, <script>
style	<base>, <head>, <html>, <meta>, <param>, <script>, <style>, <title>
title	<base>, <head>, <html>, <meta>, <param>, <script>, <style>, <title>

Tabelle 3.7: Universalattribute und die Stellen, an denen sie nicht erlaubt sind

3.2.4 Ereignisse

In die Liste der Attribute gehören auch die Ereignisse. Ereignisse sind zum Beispiel das Laden des Dokuments (`onload`), das Anklicken einer Schaltfläche (`onclick`) oder das Fahren mit der Maus über ein Element (`onmouseover`). All diese Ereignisse sind vor allem für die Arbeit mit JavaScript wichtig.

Hier ein Beispiel, das Listing 3.10 leicht variiert. Hier ist das Ereignis Klicken einer Schaltfläche zugewiesen. Das zugehörige Attribut heißt `onclick`. Sein Wert legt fest, welche JavaScript-Funktion aufgerufen wird. Hier ist es die Funktion `zoom()`, die den Text vergrößert.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
  <title>Ereignisse</title>
<script language="javascript" type="text/javascript"><!--
  function zoom() {
    document.getElementById("absatz").style.fontSize="30pt";
  }
//--></script>
</head>
<body>
  <p id="absatz">Ein Absatz mit Text.
  </p>
  <form>
    <input type="button" onclick="zoom()" value="Heranzoomen" />
  </form>
</body>
</html>
```

Listing 3.11: onclick-Attribut (onclick.html)

Hinweis

Natürlich ist dieses Beispiel nicht sehr komplex und optimiert. Es zeigt allerdings sehr gut, welch große Bedeutung Ereignisse gerade für JavaScript haben. Sie müssen sich immer überlegen, was passieren soll, um dann richtig reagieren zu können.

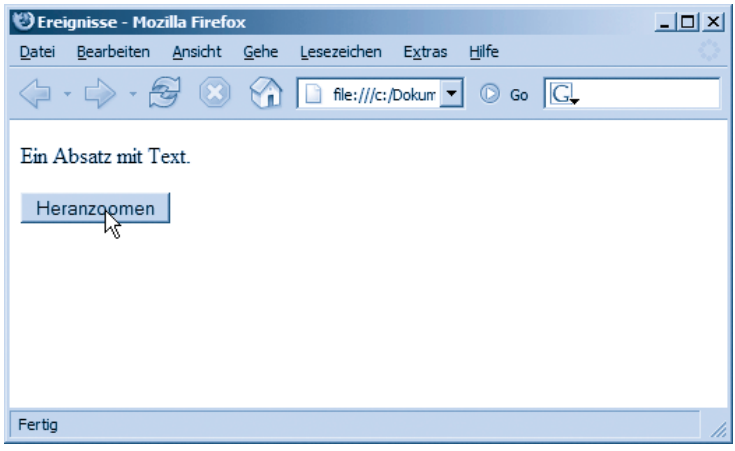


Abbildung 3.25: Erst ist der Text klein …

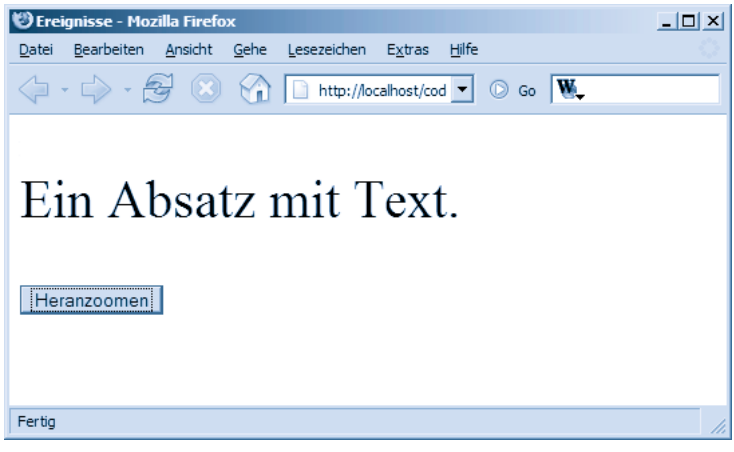


Abbildung 3.26: … und nach dem Klick ist er groß

Sie finden sie allerdings im Anhang in der Referenzübersicht aller Attribute. Dort steht auch verzeichnet, welches Ereignis für welche Tags zugelassen ist. Die dort genannten und in der W3C-HTML 4.01-Spezifikation festgehaltenen Ereignisse sind auch für alle gängigen Browser korrekt implementiert. Einzig der Netscape Navigator 4.x kennt einige Einschränkungen. Vor allem die Ereignisse, die eigentlich für alle Elemente (außer einigen logischen Ausnahmen) definiert sind, gibt es im Netscape Navigator nur für wenige Elemente. Für die wichtigsten Ereignisse enthält Tabelle 3.8 eine Kompatibilitätsübersicht.

Ereignis	Bindung NN4/And. (IE4+, Mz/Ff/NN etc.)	Beschreibung
Maus		
`onclick`	NN4: Link, Buttons And.: alles	beim Mausklick
`ondblclick`	NN4: Link And.: alles	beim Doppelklick
`onmousedown`	NN4: Link, Buttons And.: alles	beim Drücken der Maustaste
`onmouseup`	NN4: Link, Buttons, Layer And.: alles	beim Loslassen der Maustaste
`onmouseover`	NN4: Link, Bilder, Layer And.: alles	beim Bewegen des Mauszeigers über ein Element
`onmouseout`	NN4: Link, Bilder, Layer And.: alles	beim Verlassen eines Elements mit dem Mauszeiger
`onmousemove`	NN4: Event-Capturing And.: alles	beim Bewegen der Maus
Tastatur		
`onkeypress`	NN4: Textfelder And.: Textfelder	beim Drücken einer Taste
`onkeydown`	NN4: Textfelder And.: Textfelder	beim Herunterdrücken einer Taste
`onkeyup`	NN4: Textfelder And.: Textfelder	beim Loslassen einer Taste
Ändern		
`onchange`	NN4: Auswahlliste, Textfelder And.: Formularelemente	beim Ändern eines Elements
Formulare		
`onsubmit`	NN4: Formulare And.: Formulare	beim Übertragen
`onreset`	NN4: Formulare And.: Formulare	beim Zurücksetzen

Tabelle 3.8: Die Kompatibilität wichtiger Ereignisse in JavaScript[57]

57. Die Tabelle ist entlehnt aus dem »JavaScript-Kompendium« von Tobias Hauser, erschienen bei Markt+Technik. Dort finden Sie natürlich noch weiterführende Informationen über den Einsatz von Ereignissen und die unterschiedlichen Event-Modelle der Browser, die Sie für den korrekten Einsatz von JavaScript benötigen.

Ereignis	Bindung NN4/And. (IE4+, Mz/Ff/NN etc.)	Beschreibung
Fokus und Auswählen		
onfocus	NN4: Formularelemente And.: Formularelemente	beim Erhalt des Fokus
onblur	NN4: Formularelemente And.: Formularelemente	beim Verlieren des Fokus
onselect	NN4: Textfelder, File-Upload And.: Textfelder, File-Upload	beim Auswählen
Bewegen und Skalieren (meist Fenster)		
onmove	NN4: Fenster	beim Bewegen (nur Netscape Navigator 4.x)
onresize	NN4: Fenster And.: Fenster	beim Ändern der Größe
onscroll	And.: Fenster	beim Scrollen
Laden		
onload	NN4: Layer, Fenster, Bilder And.: Fenster, Frameset, Objekte	beim Laden
onunload	NN4: Fenster And.: Fenster, Frameset	beim Entladen (meist beim Verlassen der Webseite)
Fehlerhandling		
onabort	NN4: Bilder And.: Objekt	beim Abbrechen
onerror	NN4: Bilder And.: Fenster, Frameset, Objekt	bei einem Fehler

Tabelle 3.8: Die Kompatibilität wichtiger Ereignisse in JavaScript[57] (Forts.)

Hinweis

Der Internet Explorer nimmt bei den Ereignissen wie auch bei einigen anderen Themen eine Sonderrolle ein. Im Bereich der Ereignisse besitzt er wesentlich mehr als die anderen Browser. Die umfangreiche Liste finden Sie unter *http://msdn. microsoft.com/workshop/author/dhtml/reference/events.asp*.

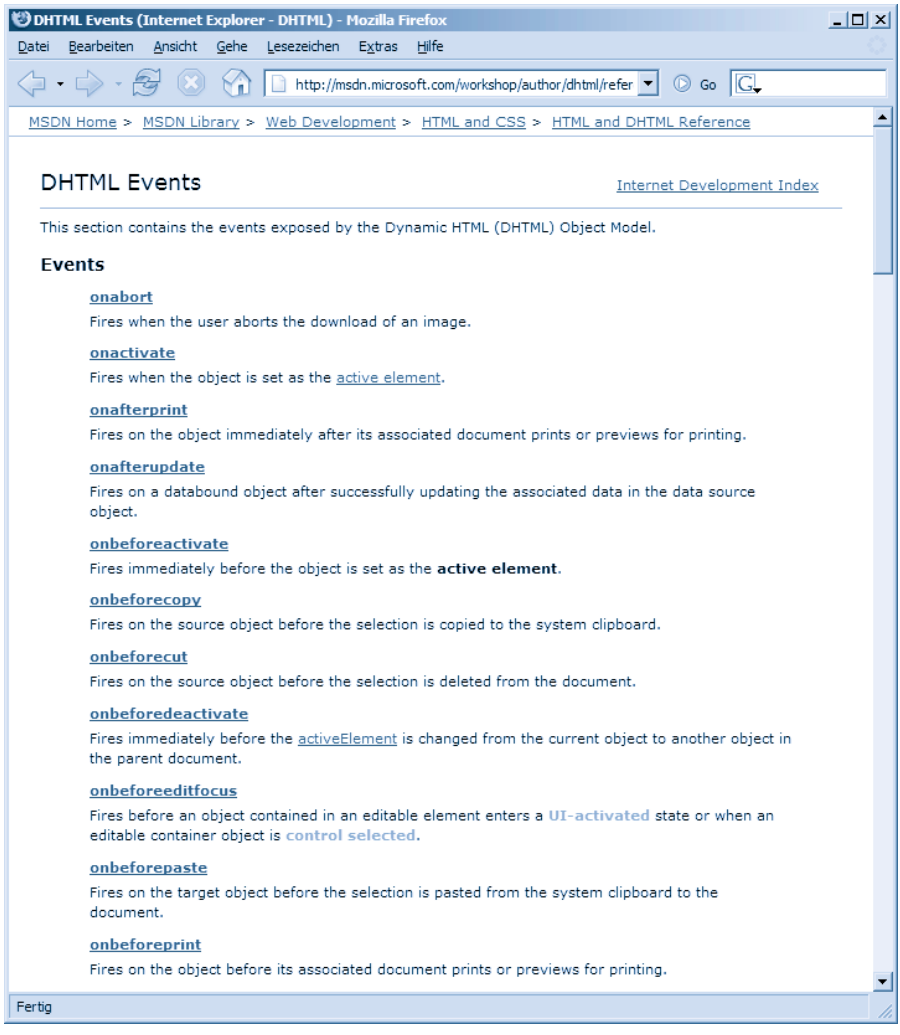


Abbildung 3.27: Zusätzliche und standardisierte Ereignisse für den Internet Explorer