## Kurzreferenz

# HTML
## (CSS, SSI, FancyIndexing, AJAX)

Autor: Michael Puff

Erstellt: 2011-10-27

http://www.michael-puff.de
mail@michael-puff.de

# Inhaltsverzeichnis

# Tabellenverzeichnis

# 1 Grundgerüst

## 1.1 Aufbau einer HTML-Datei

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <!-- Metatags -->
        <title>Seitentitel</title>
        <link rel="stylesheet" type="text/css" href="stylesheet.css"
            media="screen">
    </head>
    <body>
        <!-- Inhalt -->
    </body>
</html>
```

## 1.2 Doc typen

HTML 4.01 kennt drei Sprachvarianten:

1. `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">`
2. `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">`
3. `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">`

1. Benutzen Sie diese Angabe, wenn Sie gemäß bestimmte Elemente und Attribute nicht mehr verwenden wollen, die in früheren HTML-Standards eingefügt wurden, aber mittlerweile durch andere Möglichkeiten (vor allem durch Kapitel Stylesheets) ersetzbar sind. Ferner sind die Verschachtelungsregeln für HTML-Elemente in der Strict-Variante strenger und im Sinne strukturierter Inhalte sauberer formuliert. So ist es in dieser Variante beispielsweise nicht erlaubt, zwischen <body> und </body> einfach nur Text zu notieren. Alle Inhalte müssen in so genannten Block-Elementen stehen, z. B. in Überschriften, Textabsätzen, Tabellen usw.

2. Benutzen Sie diese Angabe, wenn Sie einige der in der Strict-Variante nicht erlaubten Elemente und Attribute verwenden wollen oder müssen. In der Variante Transitional sind zum Beispiel die Regeln für die Elementverschachtelung etwas milder. Es ist nach dieser Variante erlaubt, zwischen <body> und </body> „nackten Text“ außerhalb eines weiteren Elements zu notieren. Außerdem benötigen Sie diese Variante, wenn Sie in Seite Links mit dem target-Attribut arbeiten wollen, beispielsweise um Seite Framesets korrekt anzusteuern.
3. Diese Angabe ist nur für spezielle HTML-Dateien gedacht, in denen Framesets definiert werden.

## 1.3 Metatags

Syntax:

```
<metatag name"<bezeichner>" content="<wert>">
```

Beispiel:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1
    ">
<meta name="author" content="Michael Puff">
<meta name="robots" content="follow">
<meta name="language" content="de">
<meta name="keywords" content="Schlüsselwörter">
<meta name="description" content="Inhaltsangabe">
```

Eigenschaften, die mit *name* definiert werden, richten sich überwiegend an auslesende Client-Programme, also an Web-Browser, aber auch an Suchmaschinen-Robots, die Web-Seiten zum Füttern ihrer Suchmaschinendatenbank auslesen. Eigenschaften, die mit *http-equiv* definiert werden, sind dagegen ursprünglich für den Web-Server gedacht.

Anmerkung *refresh*:
Mit `<meta http-equiv="refresh"content="...">` wird die Weiterleitung zu einer anderen Adresse veranlasst. Mit `content="5;` wird festgelegt nach wie vielen Sekunden die Weiterleitung starten soll. Die 5 im Beispiel bedeutet also, dass die aktuelle Seite, nachdem sie geladen ist, 5 Sekunden lang angezeigt wird. Danach wird die Adresse aufgerufen, die mit `url=..."` angegeben wird.

| Metatag Name | Beschreibung |
|---|---|
| author | Autor |
| robots | Steuer das Verhalten von Suchmaschinen. *noindex*: Keine Indizierung, *index*: Indizierung erlauben, *nofollow*: Verbot Verweise zufolgen, *all*: alles erlauben. |
| language | Sprache der Seite |
| keywords | Schlüsselwörter |
| description | Kurze Beschreibung des Inhaltes |
| expires | Gültigkeitsdatum |
| date | Datum, z. B. 2001-12-15T08:49:37+02:00 |
| refresh | Weiterleitung: `<meta http-equiv="refresh" content="5; URL=http://de.selfhtml.org/">` |

Tab. 1.1: Wichtige Metatags

## 1.4 Stylesheets einbinden

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

Mögliche Werte für *media*:

1. *screen*: Stylesheet für die Bildschrimausgabe
2. *print*: Stylesheet für die Druckerausgabe

# 2 Absätze und Listen

## 2.1 Überschriften

```
<h1>Überschrift 1. Ordnung</h1>
<h2>Überschrift 2. Ordnung</h2>
<h3>Überschrift 3. Ordnung</h3>
<h4>Überschrift 4. Ordnung</h4>
<h5>Überschrift 5. Ordnung</h5>
<h6>Überschrift 6. Ordnung</h6>
```

Eine Ausrichtung ist mit dem Attribut *align* möglich. Mögliche Werte: *left*, *right*, *justify*, *center*.

## 2.2 Absätze

| Tag | Beschreibung |
|---|---|
| br | erzwungener Zeilenumbruch |
| p | Absatz |
| blockquote | Zitat, eingerückt |
| note | Anmerkung |
| banner | Fixiert einen Textblock im Verhältnis zur Seite |
| pre | vorformatierter Text |
| address | Adresse, kursiv |

Tab. 2.1: Absätze

## 2.3 Trennlinie

Mit dem Tag *hr* lässt sich eine Trennlinie erzeugen.

| Attribut | Beschreibung |
|---|---|
| width | Breite, in Pixel oder Prozent |
| size | Dicke, in Pixel |
| align | Ausrichtung |
| noshade | kein Schatten |

Tab. 2.2: Trennlinieattribute

## 2.4 Listen

### 2.4.1 Einfache Liste

```
<ul>
  <li>Eintrag</li>
  <li>Eintrag</li>
  <li>Eintrag</li>
</ul>
```

### 2.4.2 Nummerierte Liste

```
<ol>
  <li>Eintrag</li>
  <li>Eintrag</li>
  <li>Eintrag</li>
</ol>
```

| Attribut | Beschreibung |
|---|---|
| type | Legt die Art der Nummerierung fest: römisch: I oder i, alphanummerisch: A oder a |
| start | Legt einen Startwert fest |

Tab. 2.3: Listenattribute

# 3 Texthervorhebungen/ -auszeichnung

## 3.1 Logische Textbefehle

| Befehl | Beschreibung |
|---|---|
| strong | Hervorhebung, fett |
| em | allgemeine Hervorhebung, kursiv |
| cite, blockquote | Zitat |
| code | Quellcode |
| var | Variablenname, kursiv |
| samp | Beispiel |
| tt | dicktengleiche Schrift |
| kbd | Benutzereingaben |
| dfn | Definition |

Tab. 3.1: Logische Textbefehle

## 3.2 Physische Textauszeichnung

| Auszeichnung | Beschreibung |
|---|---|
| i | kursiv |
| b | fett |
| u | unterstrichen |
| s | durchgestrichen |
| sub | tiefergestellt |
| sup | hochgestellt |
| small | kleine Schrift |
| big | große Schrift |

Tab. 3.2: Physische Textauszeichnung

## 3.3 Schriftgröße

| Größe | Beschreibung |
|---|---|
| 1 | sehr klein |
| 2 | klein |
| 3 | normal |
| 4 | groß |
| 5 | größer |
| 6 | sehr groß |
| 7 | am größten |

Tab. 3.3: Schriftgröße

Beipiel: `<font size=5>Absätze als Textblöcke</font>`.

# 4 Tabellen

```
<table>
  <tr><th>Überschrift Spalte 1</th><th>Überschrift Spalte 2</th></tr>
  <tr><td>jvkbasd</td><td>jvkbasd</td></tr>
  <tr><td>jvkbasd</td><td>jvkbasd</td></tr>
  <tr><td>jvkbasd</td><td>jvkbasd</td></tr>
  <tr><td>jvkbasd</td><td>jvkbasd</td></tr>
  <tr><td>jvkbasd</td><td>jvkbasd</td></tr>
</table>
```

| Tag | Beschreibung |
|---|---|
| table | Tabelle deklarieren |
| tr | neue Zeile |
| th | Zelle für Überschrift |
| td | Zelle für Tabelleninhalt |

Tab. 4.1: Tabellen

# 5 Verweise

## 5.1 Lokale Verweise

1. Anker definieren: `<a name="<name>">Verweistext</a>`.
2. Verweis auf Anker setzen: `<a href="#<name>">Verweistext</a>`.

## 5.2 Globale Verweise

```
<a href="datei.html">Datei</a>
<a href="datei.html#ankereins">Datei - Ankereins</a>
```

# 6 Grafiken

```
<img src="speedmaster.jpg">
```

| Attribut | Beschreibung |
|---|---|
| alt | Alternativtext, wenn Grafik nicht angezeigt wird |
| title | Text der als Hinweis angezeigt wird |
| width | Breite in Pixel |
| height | Höhe in Pixel |
| border | Breite des Rahmens in Pixel |
| align | Ausrichtung. *left*, *right*, *center* |
| hspace | Horizontaler Abstand in Pixel zum Text |
| vspace | Vertikaler Abstand in Pixel zum Text |

Tab. 6.1: Grafikattribute

Anmerkung *width*/*height*: Wird nur eine Angabe gemacht, wird dasSeitenverhältnis beibehalten.

Grafiken beschriften:

```
<img src="speedmaster.jpg" align="bottom">Omega Speedmaster<p>
```

Mögliche Werte für *align*: *top*, *middle*, *bottom*.

Grafiken als Verweise:

```
<a href="http://www.omega.ch"><img src="speedmaster.jpg"></a>
```

# 7 CSS

## 7.1 Einbindung in HTML-Datei

Wird der Stylesheet in die HTML-Datei eingebettet, erfolgt dies im *head*-Abschnitt der HTML-Datei:

```
<head>
  <!-- Metatags -->
  <title>Seitentitel</title>
  <link rel="stylesheet" type="text/css" href="stylesheet.css" media="
      screen">
  <STYLE TYPE="text/css">
    H1 { color: blue }
  </STYLE>
</head>
```

Elemente können auch direkt mit Style-Angaben formatiert werden:

```
<p style="background-color:#808040; color:#D8FD02;
  font-family:'Century Schoolbook',serif; font-size:2em; letter-
      spacing:3px;
  padding:40px; border:double #D8FD02 4px;"
  title="Zitat von Francis Picabia">
  Unser Kopf ist rund, damit das Denken die Richtung wechseln kann!
</p>
```

Das Einbinden eines separaten Stylesheets erfolgt im *head*-Abschnitt der HTML-Datei:

```
<head>
  <!-- Metatags -->
  <title>Seitentitel</title>
  <link rel="stylesheet" type="text/css" href="stylesheet.css" media="
      screen">
</head>
```

## 7.2 Grundlagen

### 7.2.1 Gruppieren der Styles

Um Stylesheets möglichst klein zu halten, kann man Selektoren in Gruppen zusammenfasen, indem man sie mit mit Kommata vor der Deklaration trennt:

```
H1, H2, H3, DIV { font-family: Arial }
```

Ähnlich kann man auch mehrere Eigenschaften zusammenfassen:

```
H1 {
  font-weight: bold;
  font-size: 12px;
  line-height: 14px;
  font-family: helvetica;
}
```

### 7.2.2 Attribut *Class* als Selektor

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
<HTML>
 <HEAD>
  <TITLE>Titel</TITLE>
  <STYLE TYPE="text/css">
    H1.green { color: green }
  </STYLE>
 </HEAD>
 <BODY>
  <H1 CLASS="green">Eine grüne Überschrift</H1>
  <H1>Eine Normale Überschrift</H1>
 </BODY>
</HTML>
```

Man kann die Klassen auch für alle Elemente deklarieren:

```
.green { color: green }
```

### 7.2.3 Attribut *Id* als Selektor

Das HTML Attribut *Id* kann ebenfalls als Selektor verwand werden. Der Unterschied zu *Class* besteht darin, das ein *Id* ein Unikat in dem Dokument sein muss, also jeder Wert für *Id* darf nur einmal auftauchen. Ihnen bekommt daher eine besondere Bedeutung zu. Deklariert werden *Id* Selektoren mit einem vorangestelltem #.

```
#Wide { letter-spacing: 0.3em }
H1#Wide { letter-spacing: 0.5em }
```

### 7.2.4 Kontextabhängige Selektoren

Die Vererbung von Eigenschaften erspart einem viel Tipparbeit. Man definiert Standardeigenschaften und behandelt Ausnahmen separat. Wenn man z.B. dem Element EM innerhalb von H1 ein besonderes Aussehen verliehen will, kann man schreiben:

```
H1 { color: blue }
EM { color: red }
```

Jetzt wird das Element EM innerhalb von H1 wie gewollt rot dargestellt, aber überall sonst im Dokument auch. Um das zu verhindern kann man die Styles abhängig von ihrem Auftauchen deklarieren.

```
H1 EM { color: red }
```

Wenn jetzt der Hierarchie entsprechend ein EM innerhalb von H1 gefunden wird, so wird das rot dargestellt, sonst nicht.

Das ganze kann man dann auch noch weiter verschachteln, z.B.:

```
DIV H1 EM { color: #ff00ff }
```

Kontextabhängige Selektoren können von Elementen, Class Attributen, Id Attributen oder Kombinationen daraus abhängig gemacht werden, z.B.:

```
DIV P           { font: small sans-serif }
.green H1       { color: blue }
#x78y CODE      { background: blue }
DIV.sidenote H1 { font-size: large }
```

Bedeutet soviel wie: P in DIV bekommt kleine sans-serif Schrift; H1 innerhalb von `Class=green` wird blau gefärbt; CODE innerhalb von `ID="#x78y"` bekommt blauen Hintergrund; H1 innerhalb von DIV mit `Class=sidenote` wird groß dargestellt.

Ebenso wie auch normale Selektoren, können kontextabhängige Selektoren gruppiert werden:

```
H1 B, H2 B, H1 EM, H2 EM { color: red }
```

Was dasselbe ist, wie:

```
H1 B  { color: red }
H2 B  { color: red }
H1 EM { color: red }
H2 EM { color: red }
```

### 7.2.5 Kommentare

Kommentare in CSS sind denen in Programmiersprachen wie C sehr ähnlich:

```
A { color: green } /* Linkfarbe ist grün */
```

CSS Kommenatare können nicht ineinander verschachtelt werden, und Kommentare wie man sie von JavaScript her kennt ala `// Kommentar` sind auch nicht zulässig.

## 7.3 CSS-Eigenschaften

### 7.3.1 Schriftformatierung

**font-family** Mit *font-family:* kann eine oder mehrere Schriftarten festgelegt werden. Bei mehreren angegebenen Schriftarten ist die Reihenfolge der Angabe entscheidend: ist die erste angegebene Schriftart verfügbar, wird diese verwendet. Ist sie nicht verfügbar, wird die zweite Schriftart verwendet, falls diese verfügbar ist usw. Trennen Sie mehrere Schriftartennamen durch Kommata. Das W3-Konsortium empfiehlt, Schriftartnamen, die Leerzeichen enthalten, in Anführungszeichen zu setzen. Es

empfiehlt sich, generische Schriftarten als letzte Angabe einer Wertzuweisung an font-family zu notieren.

- *serif* = eine Schriftart mit Serifen,
- *sans-serif* = eine Schriftart ohne Serifen,
- *cursive* = eine Schriftart für Schreibschrift,
- *fantasy* = eine Schriftart für ungewöhnliche Schrift,
- *monospace* = eine Schriftart mit dicktengleichen Zeichen.

**font-size** Mit *font-size:* kann die Schriftgröße festgelegt werden. Erlaubt ist eine numerische Angabe. Auch Prozentangaben sind erlaubt. Prozentwerte beziehen sich auf die Schriftgröße des Elternelements. Alternativ zu numerischen Angaben sind auch absolute und relative Schlüsselworte möglich.
Absolut:

- *xx-small* = winzig.
- *x-small* = sehr klein.
- *small* = klein.
- *medium* = mittel.
- *large* = groß.
- *x-large* = sehr groß.
- *xx-large* = riesig.

Relativ:

- *smaller* = kleiner als im Elternelement.
- *larger* = größer als im Elternelement.

Für die Ausgabe auf dem Bildschirm sind vor allem relative Einheiten wie *em*, *ex*, Prozentwerte oder Schlüsselworte geeignet, absolute Einheiten wie *pt*, *cm* usw. sollten Drucklayouts vorbehalten sein.

**font-style** Mit *font-style:* kann der Schriftstil bestimmt werden. Folgende Angaben sind möglich:

- *italic* = kursiver Schriftstil.
- *oblique* = schräggestellter Schriftstil.
- *normal* = normaler Schriftstil.

**font-variant** Mit *font-variant:* kann die Schriftvariante definiert werden. Folgende Angaben sind möglich:

- *small-caps* = Kapitälchen.
- *normal* = normale Schriftvariante.

**font-weight** Mit *font-weight:* kann das Schriftgewicht bestimmt werden. Folgende Angaben sind möglich:

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

Bei den numerischen Werten entspricht die Angabe 500 dem im DTP-Bereich üblichen Begriff *medium*, und die Angabe 700 entspricht dem Begriff *bold*.

**text-decoration** Mit *text-decoration:* kann die zusätzliche Formatierung festgelegt werden. Folgende Angaben sind möglich:

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

**color** Mit color: kann die Textfarbe festgelegt werden.

### 7.3.2 Ausrichtung und Absatzkontrolle

**text-indent** Mit *text-indent:* kann eine Einrückung für die erste Zeile festgelegt werden. Erlaubt ist eine numerische Angabe.

**line-height** Mit *line-height:* kann die Zeilenhöhe definiert werden. Erlaubt ist eine numerische Angabe. Auch Prozentangaben sind erlaubt. Prozentwerte beziehen sich dabei auf die Schriftgröße des Elements, für das die Zeilenhöhe bestimmt wird.

**vertical-align** Mit *vertical-align:* die vertikale Ausrichtung festgelegt werden. Folgende Angaben sind möglich:

- *top* = obenbündig ausrichten.
- *middle* = mittig ausrichten.
- *bottom* = untenbündig ausrichten.
- *baseline* = an der Basislinie ausrichten (oder untenbündig, wenn es keine Basislinie gibt).

- *sub* = tieferstellen (ohne die Schriftgröße zu reduzieren).
- *super* = höherstellen (ohne die Schriftgröße zu reduzieren).
- *text-top* = am oberen Schriftrand ausrichten.
- *text-bottom* = am unteren Schriftrand ausrichten.

**text-align** Mit *text-align:* können Textinhalte von Block-Elementen ausgerichtet werden. Folgende Angaben sind möglich:

- *left* = linksbündig ausrichten (Voreinstellung).
- *center* = zentriert ausrichten.
- *right* = rechtsbündig ausrichten.
- *justify* = als Blocksatz ausrichten.

### 7.3.3 Ränder

Außenrand oder Abstand bedeutet: erzwungener Leerraum zwischen dem aktuellen Element und seinem Elternelement oder Nachbarelement. Für ein p-Element, also einen Textabsatz etwa, der direkt innerhalb des body-Elements notiert ist, markieren Angaben zu linkem und rechten Außenrand seinen Abstand zu den Elementgrenzen des body-Elements. Wenn mehrere solcher p-Absätze aufeinander folgen, markieren Angaben zum Außenrand oben und unten den Abstand zwischen den Absätzen.

- *margin-top* = Abstand oben
- *margin-right* = Abstand rechts
- *margin-bottom* = Abstand unten
- *margin-left* = Abstand links

### 7.3.4 Rahmen

**border-width** Mit *border-width:* kann die Dicke des Rahmens um ein Element festgelegt werden. Erlaubt ist eine Seite numerische Angabe (mit Ausnahme von Prozentwerten) für die Rahmendicke oder einer der folgenden Werte:

- *thin* = dünn.
- *medium* = mittelstark.
- *thick* = dick.

Die Angabe *border-width* wird nur interpretiert, wenn außerdem der Rahmentyp *border-style* angegeben wird.

**border-color** Mit *border-color:* wird die Rahmenfarbe festgelegt. Erlaubt ist eine Farbangabe oder der Wert *transparent*.

**border-style** Mit *border-style:* wird der Rahmentyp festlegt. Erlaubt ist eine der folgenden Angaben.

- *none* = kein Rahmen (bzw. unsichtbarer Rahmen).
- *hidden* = kein Rahmen (bzw. unsichtbarer Rahmen)
- *dotted* = gepunktet.
- *dashed* = gestrichelt.
- *solid* = durchgezogen.
- *double* = doppelt durchgezogen.
- *groove* = 3D-Effekt.
- *ridge* = 3D-Effekt.
- *inset* = 3D-Effekt.
- *outset* = 3D-Effekt.

### 7.3.5 Innenabstand

- *padding-top* = Innenabstand oben
- *padding-right* = Innenabstand rechts
- *padding-bottom* = Innenabstand unten
- *padding-left* = Innenabstand links
- *padding* = Innenabstand allgemein

### 7.3.6 Hintergrundfarben und -bilder

**background-color** Mit *background-color:* kann eine Hintergrundfarbe festgelegt werden. Erlaubt ist eine Farbangabe oder der Default-Wert *transparent*.

**background-image** Mit *background-image:url([URI])* kann eine Hintergrundgrafik festgelegt werden.

**background-repeat** Mit *background-repeat:* kann das Wiederholungsverhalten einer Hintergrundgrafik, die mit *background-image* eingebunden wurde, kontrolliert werden. Erlaubt ist eine der folgenden Angaben.

- *repeat* = wiederholen (Voreinstellung).
- *repeat-x* = nur „eine Zeile lang“ waagerecht wiederholen.
- *repeat-y* = nur „eine Spalte lang“ senkrecht wiederholen.

- *no-repeat* = nicht wiederholen, nur als Einzelbild anzeigen.

**background-attachment** Mit *background-attachment:* kann das Scroll-Verhalten einer Hintergrundgrafik, die mit *background-image* eingebunden wurde, kontrolliert werden. Erlaubt ist eine der folgenden Angaben.

- *scroll* = mitscrollen (Voreinstellung), orientiert sich an der Position des jeweiligen Elements
- *fixed* = Hintergrundbild bleibt stehen, orientiert sich am Viewport

**background-position** Mit *background-position:* kann festgelegt werden, wo die linke obere Ecke der Hintergrundgrafik sein soll. Der erste Wert steht für die horizontale, der zweite für die vertikale Position. Bezugspunkt ist das HTML-Element, für das die Hintergrundgrafik definiert wird. Etwaige Innenabstände werden berücksichtigt. Erlaubt sind numerische Angaben und folgende Angaben:

- *top* = vertikal obenbündig.
- *bottom* = vertikal untenbündig.
- *center* = zentriert (horizontal oder vertikal, center center kann als center zusammengefasst werden).
- *left* = horizontal linksbündig.
- *right* = horizontal rechtsbündig.

### 7.3.7 Listenformatierung

**list-style-type** Mit *list-style-type:* kann das Aussehen von Listenzeichen kontrolliert werden. Erlaubt ist eine der folgenden Angaben.

- *decimal* = für *ol*-Listen: Nummerierung 1.,2.,3.,4. usw.
- *lower-roman* = für *ol*-Listen: Nummerierung i.,ii.,iii.,iv. usw.
- *upper-roman* = für *ol*-Listen: Nummerierung I.,II.,III.,IV. usw.
- *lower-alpha* oder *lower-latin* = für *ol*-Listen: Nummerierung a.,b.,c.,d. usw.
- *upper-alpha* oder *upper-latin* = für *ol*-Listen: Nummerierung A.,B.,C.,D. usw.
- *disc* = für *ul*-Listen: gefüllter Kreis als Bullet-Zeichen
- *circle* = für *ul*-Listen: leerer Kreis als Bullet-Zeichen
- *square* = für *ul*-Listen: rechteckiges Bullet-Zeichen
- *none* = kein Bullet-Zeichen, keine Nummerierung

**list-style-position** Mit *list-style-position:* kann das Einrückungsverhalten festgelegt werden. Erlaubt ist eine der folgenden Angaben.

- *inside* = eingerückt.
- *outside* = ausgerückt (Voreinstellung).

**list-style-image** Mit *list-style-image:url([Dateiname])* kann eine Grafik für das eigene Bullet-Zeichen angeben werden.

### 7.3.8 Tabellenformatierung

**caption-side** Mit *caption-side:* kann die Position der Tabellenüberschrift bestimmt werden. Folgende Angaben sind erlaubt:

- *top* = oberhalb der Tabelle.
- *bottom* = unterhalb der Tabelle.

**table-layout** *table-layout:* beeinflusst die Tabellenanzeige bei Breitenangaben. Folgende Angaben sind erlaubt:

- *auto* = Zelleninhalt hat Vorrang vor Breitenangaben (Voreinstellung).
- *fixed* = Breitenangaben haben Vorrang vor dem Zelleninhalt.

Der Einsatz des Wertes *fixed* hat zur Folge, dass sich die Zellenbreite nicht wie herkömmlich an deren Inhalt orientiert, sondern an der mittels *width* explizit vorgegebenen Breite der Spalten der ersten Zeile. Wurde für eine oder mehrere Spalten keine Breite definiert, wird noch verfügbarer Platz auf diese aufgeteilt. Wurden alle Spalten mit einer Breite versehen und ist für die Tabelle insgesamt eine größere Breite definiert, als sich aus der Addition aller Spaltenbreiten ergibt, wird zusätzlich vorhandener Platz unter Berücksichtigung von Rahmenbreiten und Zellabständen zu gleichen Teilen auf alle Spalten aufgeteilt. Wie ein für eine Zelle zu breiter Inhalt dargestellt wird, ergibt sich aus dem Wert der Eigenschaft *overflow*.

**border-collapse** Mit *border-collapse:* beeinflusst die Art, wie Einzelrahmen benachbarter Tabellenzellen reagieren. Folgende Angaben sind erlaubt:

- *separate* = Zellenrahmen fallen nicht zusammen (Ausgangswert seit CSS 2.1).
- *collapse* = Zellenrahmen fallen zusammen (Ausgangswert in CSS2).

**border-spacing** Mit *border-spacing:* kann für ein *table*-Element den Abstand der Zellenrahmen voneinander bestimmt werden. Erlaubt ist eine numerische Angabe.

**empty-cells** Mit *empty-cells:* wird bestimmt, ob Rahmen leerer Tabellenzellen angezeigt werden oder nicht. Folgende Angaben sind erlaubt:

- *show* = Zellenrahmen leerer Tabellenzellen werden angezeigt.
- *hide* = Zellenrahmen leerer Tabellenzellen werden unterdrückt (Voreinstellung).

### 7.3.9 Positionierung und Anzeige von Elementen

**position** Mit *position:* kanndie Positionsart bestimmt werden. Folgende Angaben sind erlaubt:

- *static* = keine spezielle Positionierung, normaler Elementfluss (Normaleinstellung).
- *relative* = relative Positionierung (Verschiebung), gemessen an der Normalposition oder Anfangsposition des Elements selbst.
- *absolute* = absolute Positionierung, gemessen am Rand des nächsthöheren Vorfahrenelements, das nicht die Normaleinstellung position:static aufweist. Scrollt mit.
- *fixed* = absolute Positionierung, gemessen am „Viewport“, d. h. am Browserfenster. Bleibt beim Scrollen stehen.

**Startpositionen** Diese Angabe ist sinnvoll in Verbindung mit einer vom Wert *static* abweichenden Angabe zu *position*. Es kann bestimmt werden, wo ein absolut oder relativ positioniertes Element beginnt.

- *top* = Startposition von oben
- *left* = Startposition von links
- *bottom* = Startposition von unten
- *right* = Startposition von rechts

**width** Mit *width:* kann die Breite bestimmt werden. Erlaubt ist eine numerische Angabe oder der Wert *auto* für automatische Breite.

- *min-width* = Mindestbreite
- *max-width* = Maximalbreite

**height** Mit *height:* kann die Höhe festgelegt werden. Erlaubt ist eine numerische Angabe oder der Wert *auto* für automatische Höhe.

- *min-height* = Mindesthöhe
- *max-height* = Maximalhöhe

**float** Mit *float:* kann bestimmt werden, dass nachfolgende Elemente das aktuelle Element bzw. den aktuellen Bereich umfließen. Folgende Angaben sind möglich:

- *left* = Element steht links und wird rechts davon von nachfolgenden Elementen umflossen.
- *right* = Element steht rechts und wird links davon von nachfolgenden Elementen umflossen.
- *none* = Kein Umfluss (Normaleinstellung).

**clear** Mit *clear:* kann ein Umfluss abgebrochen und die Fortsetzung unterhalb des umflossenen Elements oder Bereichs erzwungen werden. Folgende Angaben sind möglich:

- *left* = Erzwingt bei float:left die Fortsetzung unterhalb.
- *right* = Erzwingt bei float:right die Fortsetzung unterhalb.
- *both* = Erzwingt in jedem Fall die Fortsetzung unterhalb.
- *none* = Erzwingt keine Fortsetzung unterhalb. (Normaleinstellung).

**display** Mit *display:* kann die Anzeige von Elementen unterdrücken oder die Art der Anzeige festlegt werden. Folgende Angaben sind möglich:

- *block* = Erzwingt einen Block - das Element erzeugt eine neue Zeile.
- *inline* = Erzwingt die Anzeige im Text - das Element wird im laufenden Textfluss angezeigt.
- *inline-block* = Erzeugt äußerlich einen Block, für den Breite, Höhe und Außenabstand angegeben werden kann, belässt das Element jedoch im laufenden Textfluss - ähnlich einem „inline replaced element“ wie *img*. Dieser Wert wird erst mit CSS 2.1 eingeführt.
- *list-item* = wie block, jedoch mit einem Aufzählungszeichen (Bullet) davor.
- *run-in* = bewirkt, dass das Element kontext-abhängig als Block-Element oder als Inline-Element dargestellt wird.
- *none* = Element wird nicht angezeigt und es wird auch kein Platzhalter freigelassen.

**visibility** Mit *visibility:* kann bestimmt werden, ob ein Element anfangs angezeigt wird oder nicht. Folgende Angaben sind möglich:

- *visible* = Der Inhalt des Element wird angezeigt (Normaleinstellung).

- *hidden* = Der Inhalt des Element wird versteckt (nicht angezeigt).
- *collapse* = Spalten oder Reihen einer Tabelle werden versteckt und geben den zuvor benötigten Platz frei. Wirkt auf alle anderen Elemente wie *hidden*.

**clip** Mit *clip:* kann ein Ausschnitt für die sichtbare Anzeige definiert werden. Dahinter folgt der Bezeichner *rect* (für rectangle, = Rechteck), und dahinter, in runde Klammern eingeschlossen, vier durch Beistriche getrennte numerische Werte oder das Schlüsselwort *auto* zur Bestimmung des Ausschnitts; `clip:rect(0px, 130px, 130px, 0px)`

- Der erste der vier Werte bezeichnet den Wert für „oben“, gemessen an der oberen Elementgrenze
- Der zweite der vier Werte bezeichnet den Wert für „rechts“, gemessen an der linken Elementgrenze
- Der dritte der vier Werte bezeichnet den Wert für „unten“, gemessen an der oberen Elementgrenze
- Der vierte der vier Werte bezeichnet den Wert für „links“, gemessen an der linken Elementgrenze

### 7.3.10 Pseudoelemente und Pseudoklassen

**:link, :visited, :focus, :hover, :active** Die beschriebenen Pseudoklassen gelten teilweise nur für das *a*-Element in HTML, daher wird vor dem Doppelpunkt das *a* notiert. In den Formatdefinitionen für die einzelnen Pseudoklassen können beliebige, geeignete CSS-Eigenschaften zugewiesen werden. Die folgenden Pseudoklassen bedeuten:

- *:link* = für Verweise zu noch nicht besuchten Seiten
- *:visited* = für Verweise zu bereits besuchten Seiten
- *:focus* = für Elemente, die den Fokus erhalten, z. B. durch „Durchsteppen“ mit der Tabulator-Taste (CSS 2.0)
- *:hover* = für Elemente, während der Anwender mit der Maus darüber fährt (CSS 2.0)
- *:active* = für gerade angeklickte Elemente

**:first-child, :first-line, :first-letter** Die beschriebenen Pseudoelemente und Pseudoklassen sind für typische Absatz- oder Überschriftenelemente in HTML gedacht. In den Deklarationen können beliebige, geeignete CSS-Eigenschaften zugewiesen werden. Die Pseudoelemente und Pseudoklassen bedeuten:

- *:first-child* = das Element, das das erste Kindelement eines anderen Elements ist erhält die CSS-Eigenschaften
- *:first-line* = die erste Textzeile des Elements erhält die CSS-Eigenschaften
- *:first-letter* = das erste Zeichen des Textes erhält die CSS-Eigenschaften

# 8 Server Side Includes

**Was bedeutet SSI?**
Server Side Includes beschreibt einen Vorgang bei dem der Webserver vor dem ausliefern der Seite noch Code oder Informationen in die Seite einfügt. Die Dateien müssen die Endung shtml haben, damit der Server erkennt, dass er die Seite noch parsen und eventuell noch Code oder Informationen einfügen muss.

**Allgemeine Syntax**

```
<!--#befehl variablenname="wert"-->
```

**Zeit ausgeben**

```
<!--#echo var="date_gmt"-->
```

Zeitangabe formatieren:

```
{<!--#config timefmt="%A, %d %B %Y at %H:%M:%S"-->
```

ergibt *Wednesday, 26 August 2009 um 10:17:06*.

**Dateigröße ausgeben**

```
<!--#fsize file="index.shtml"-->
```

Dateigröße formatieren:

```
<!--#config sizefmt="bytes"-->
```

Mögliche Werte für *sizefmt*: bytes, kbytes, mbytes, abbrev. *abbrev* fügt die Abkürzung der Einheit an.

**Dateidatum**
Datum der letzten Änderung einer bestimmten Datei:

```
<!--#flastmod file="index.shtml"-->
```

Datum der letzten Änderung der aktuellen Datei:

| Element | Ausgabe | Beispiel |
|---|---|---|
| %a | Abkürzung des Wochentages | Sun |
| %A | Wochentag | Sunday |
| %b | Monat (abk.) | Jan |
| %B | Monat | January |
| %d | Datum | 1 (and not 01) |
| %H | Uhrzeit (24h) | 13 |
| %I | Uhrzeit (12h) | 1 |
| %j | Tag des Jahres | 360 |
| %m | Monat | 11 |
| %M | Minuten | 08 |
| %p | AM oder PM | AM |
| %S | Sekunden | 09 |
| %U | Woche des Jahres(auch %W) | 49 |
| %w | Tag der Woche | 05 |
| %y | Jahr (JJ) | 04 |
| %Y | Jahr (JJJJ) | 1995 |
| %Z | Zeitzone | EST |
| %z | Abweichung von GMT | |

Tab. 8.1: SSI Zeitformate

```
<!--#echo var="LAST_MODIFIED"-->
```

### Dateien einbinden

```
<!--#include file="datei.shtml"-->
```

Es können sowohl Nur-Text als auch HTML formatierte Dateien eingebunden werden. Allerdings könne keine PHP-Dateien eingebunden werden.

### Verzweigung

```
<!--#config timefmt="%A"-->
<!--#if expr="$date_gmt = Friday" -->
Yippie! Bald ist Wochenende
<!--#elif expr="($date_gmt = Saturday) || ($date_gmt = Sunday)" -->
Wochenende!
<!--#else -->
Schoener Tag heute!
<!--#endif -->
```

### Variablen

```
<!--#set var="fahrzeug" value="Mercedes" -->
<!--#echo var="fahrzeug" --><BR>
<!--#if expr="$fahrzeug = Mercedes" -->
Nettes Auto!
<!--#endif -->
```

### Ausgabe

Mit *echo* können Browser und Server Informationen (Umgebungs Variablen) ausgegeben werden.

| Variable | Code |
|---|---|
| Document Name | `<!--#echo var="document_name"-->` |
| Document URI | `<!--#echo var="document_uri"-->` |
| Local Date | `<!--#echo var="date_local"-->` |
| GMT Date | `<!--#echo var="date_gmt"-->` |
| Last Modified | `<!--#echo var="last_modified"-->` |
| Server Software | `<!--#echo var="server_software"-->` |
| Server Name | `<!--#echo var="server_name"-->` |
| Server Protocol | `<!--#echo var="server_protocol"-->` |
| Server Port | `<!--#echo var="server_port"-->` |
| Gateway Interface | `<!--#echo var="gateway_interface"-->` |
| Request Method | `<!--#echo var="request_method"-->` |
| Script Name | `<!--#echo var="script_name"-->` |
| Remote Host | `<!--#echo var="remote_host"-->` |
| Remote Address | `<!--#echo var="remote_addr"-->` |
| Remote User | `<!--#echo var="remote_user"-->` |
| Content Type | `<!--#echo var="content_type"-->` |
| Content Length | `<!--#echo var="content_length"-->` |
| HTTP Accept | `<!--#echo var="http_accept"-->` |
| HTTP User Agent (Browser) | `<!--#echo var="http_user_agent"-->` |
| HTTP Cookie | `<!--#echo var="http_cookie"-->` |
| Unescaped query string | `<!--#echo var="query_string_unescaped"-->` |
| Query String | `<!--#echo var="query_string"-->` |
| Path Info | `<!--#echo var="path_info"-->` |
| Path Translated | `<!--#echo var="path_translated"-->` |
| Referer | `<!--#echo var="referer"-->` |
| Angeforderte URI | `<!--#echo var="request_uri"-->` |
| Forwarded | `<!--#echo var="forwarded"-->` |

Tab. 8.2: SSI Umgebungsvariablen

### Skripte ausführen

```
<!--#exec cgi="../cgi-bin/forum.cgi"-->
```

# 9 Directory-Listings des Apache anpassen - FancyIndexing

Es ist mit den .htaccess-Dateien möglich das Aussehen der Standardseiten z. B. das Aussehen der Directory-Listings zu beeinflussen.

## 9.1 Vorraussetzungen

Grund Voraussetzung, dass die hier beschriebene Technik funktioniert, ist, dass das DirectoryListing aktiviert ist. Dies kann man entweder im Controlpanel seines Webspaces einstellen oder man gibt in der .htacces Datei als erstes die Option:

```
Options +Indexes
```

an. Allgemein kann man sich merken, dass ein „+“ eine Option aktiviert und ein „-“ eine Option deaktiviert.

| Option | Beschreibung |
|---|---|
| ExecCGI | Steuert das ausführen von CGI-Skripten in dem betreffenden Verzeichnis. |
| Includes | Steuert das parsen von Server-Sided-Includes [SSI]. |
| IncludesNOEXEC | Steuert, ob Include-Dateien CGI-Skripte ausführen dürfen. Wirkt sich entsprechend auf die #exec- und #include-Tags aus. |
| Indexes | Steuert, ob der Server einen Verzeichnis Inhalt zurückliefern soll, wenn keine Index-Datei gefunden wurde oder nicht. |

Tab. 9.1: .htaccess Optionen

Desweiteren muss in der .htacces-Datei mit `FancyIndexing On` das FancyIndexing eingeschaltet werden.

## 9.2 Aussehen der Verzeichnistabelle anpassen

Gesteuert wird das grundsätzliche Aussehen des DirectoryListings mit dem Schalter: IndexOptions. Die möglichen und gewünschten Optionen werden dahinter einfach angegeben:

| **Parameter** | **Beschreibung** |
|---|---|
| IconHeight=[Wert in Pixel] | Höhe des Icons in Pixel. |
| IconWidth=[Wert in Pixel] | Breite des Icons in Pixel. |
| IconsAreLinks | Bestimmt, ob die Icons Links sind. Allerdings haben sie dann immer eine Rand, was eventuell nicht so schön aussieht. |
| NameWidth=[n*] | Breite der Spalte Name in Zeichen. Ein „*“ setzt die Breite auf den längsten Dateinamen. |
| ScanHTMLTitles | Ist diese Optiion angegeben, wird der Titel der HTML-Datei dazu benutzt, um eine Beschreibung zu generieren. |
| SuppressColumnSorting | Die Spalten können nicht sortiert werden. |
| SuppressDescription | Die Spalte *Beschreibung* wird unterdrückt. |
| SuppressLastModified | Die Spalte *Letzte Änderung* wird unterdrückt. |
| SuppressSize | Die Spalte *Dateigröße* wird unterdrückt. |
| SuppressHTMLPreamble | Unterdrückt die Erzeugung des HTML-Headers einer HTML-Datei. Sollte man machen, wenn man einen extra Header angibt. |
| FoldersFirst | Ordner werden als erstes aufgelistet. |

Tab. 9.2: Parameter FancyIndexing IndexOptions

```
IndexOptions NameWidth=30 DescriptionWidth=450 IconHeight=16 IconWidth
    =16
    SuppressHTMLPreamble FoldersFirst ScanHTMLTitles
```

## 9.3 Eigene Icons für die Dateitypen

Der Apache nimmt für jeden Dateityp ein passendes Icon. Nur leider sehen diese etwas ünschönäus. Dies wollen wir ändern. Mit der Option *AddIcon* können wir einem Dateitypen ein Icon zuweisen. Wir haben dabei die Möglichkeit Wildcards, wie wir sie schon von DOS kennen, zu benutzen. Folgende Zeile weist dem Dateityp *.html das mit dem Pfad angegebene Bildchen zu:

```
AddIcon /pics/html.png *.html *.shtml
```

Wie man sehen kann, kann man auch mehrere Dateitypen angeben durch ein Leerzeichen getrennt. Mit der Option DefaultIcon kann für alle restlichen Dateitypen ein Icon bestimmt werden.

Gehen wir gleich noch einen Schritt weiter. Laut dem W3C sollte jede Grafik auch einen alternativen Text anzeigen. Auch diese Möglichkeit bietet uns FancyIndexing. Mit der Option *AddAlt* kann jedem Icon neben einem Dateinamen ein alternativer Text zugewiesen werden. Dabei muss der Text in Hochkommas und der Dateityp angegeben werden:

```
AddAlt "Webseite" html shtml
```

Fährt man nun mit der Maus über das Icon vor einem Dateinamen wird die Beschreibung in einem kleinen Hinweisfensterchen angezeigt.

Bleiben noch zwei Sonderfälle. Der erste wäre ein Icon für ein Verzeichnis. Jetzt könnte man bei *AddIcon* jeden Ordner einzeln angeben, was allerdings etwas mühsam wäre. Gibt man statt dessen `^^DIRECTORY^^` als Dateityp an, wird jeder Ordner mit dem angegebenen Icon angezeigt.

```
AddIcon /pics/folder.png ^^DIRECTORY^^
```

Dann gibt es da noch das unsichtbare Icon am Anfang des Tabellenkopfes. Es dient als Platzhalter damit die Ausrichtung der Tabellenspalten stimmt. Dieses können wir mit der Angabe `^^BLANKICON^^` ändern.

## 9.4 Beschreibungen für Dateien und Ordner angeben

Die Vorgehensweise ist ähnlich wie die Angabe eines alternativen Textes bei den Dateitypen, nur dass die Angabe *AddDescription* benutzt wird. In den

Hochkommas steht die Beschreibung und anschließend die Datei oder Ordner. Auch hier sind wieder Wildcards und die Angabe mehrer Dateien erlaubt.

```
AddDescription "Support-Forum" forum
```

Es ist nicht möglich innerhalb der Beschreibung mit weiteren HTML-Tags wie Tags für Links oder Grafiken zu arbeiten.

## 9.5 Dateien für den Header und Footer

Man kann auch über und unter dem DirectoyListing eigene HTML-Dateien anzeigen lassen wenn einem der Standardtext vom Apache nicht gefällt. Verantwortlich dafür sind die Einstellungen: HeaderName und ReadmeName

```
HeaderName /data/head.html
ReadmeName /data/foot.html
```

Hier wird über dem DirectoryListing die Datei *head.html* und unter dem Listing die Datei *foot.html* angezeigt. Es kann sich dabei auch um shtml-Dateien handeln mit zusätzlichen Includes. Hier kommt ist es jetzt auch wichtig bei den IndexOptions *SuppressHTMLPreamble* anzugeben, um eine gültige HTML-Datei vom Server geliefert zu bekommen.

## 9.6 Dateien und Ordner ausblenden

Hat man das DirectoryListing aktiviert ist es eventuell unerwünscht, dass ein Besucher bestimmte Dateien und Ordner sieht, um nicht in Verlegenheit zu kommen diese einzusehen. FancyIndexing biete da die Möglichkeit diese Ordner und Dateien auszublenden beim DirectoryListing. Folgende Angabe:

```
IndexIgnore cgi-bin error includes pics programme css *.php *.ico *.
    png
```

blendet die Verzeichnisse *cgi-bin*, *error*, *includes*, *pics* und *programnme* und die Dateien mit der Endung *php* und *ico* aus.

# 10 Entities

| Zeichen | HTML Entity |
|---|---|
| „ | &quot; |
| & | &amp; |
| < | &lt; |
| > | &gt; |
| © | &copy; |
| Ä, ä | Ä, ä |
| Ö, ö | Ö, ö |
| Ü, ü | Ü, ü |
| ß | &szlig; |
| Leerzeichen |   |
| Euro-Zeichen | € |

Tab. 10.1: Wichtige Entities

Weitere Sonderzeichen: http://de.selfhtml.org/html/referenz/zeichen.htm

# 11 Seitenvorlage

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


<html xmlns="http://www.w3.org/1999/xhtml" dir="ltr" lang="de">
    <head>
      <meta http-equiv="content-type" content="text/html; charset=UTF
          -8" />
      <meta http-equiv="Content-Style-Type" content="text/css" />
      <meta name="author" content="Michael Puff" />
      <meta name="robots" content="follow" />
      <meta name="language" content="de" />
      <meta name="keywords" content="" />
      <meta name="description" content="" />
      <link rel="stylesheet" type="text/css" href="/data/stylesheet.
          css" media="screen" />
      <link rel="stylesheet" type="text/css" href="/data/print.css"
          media="print" />
      <title>Seitenvorlage</title>
    </head>
    <body>
      <!--#include virtual="/data/nav.html" -->
      <h1>Seitenvorlage</h1>
      <h2 style="text-align:center;">Zweite Überschrift zentriert</h2>
      <p>
        Inhalt.
      </p>
<!--#include virtual="/data/footer.shtml" -->
```

# 12 AJAX

## 12.1 Was ist AJAX?

Ajax ist ein Apronym für die Wortfolge „Asynchronous JavaScript and XML“. Es bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Browser und dem Server. Dieses ermöglicht es, HTTP-Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden [3]. Siehe dazu die Grafik auf Seite 36.

## 12.2 Verwendete Technologien

- HTML (oder XHTML)
- Document Object Model (DOM) zur Repräsentation der Daten oder Inhalte
- JavaScript zur Manipulation des Document Object Models und zur dynamischen Darstellung der Inhalte. JavaScript dient auch als Schnittstelle zwischen einzelnen Komponenten.
- Das XMLHttpRequest-Objekt, Bestandteil vieler Browser, um Daten auf asynchroner Basis mit dem Webserver austauschen zu können.
- JSON, ein auf JavaScript zugeschnittenes, textbasiertes Format für Daten und Objekte.
- SOAP, ein Protokoll für Webservices, das meist XML als Austauschformat verwendet.

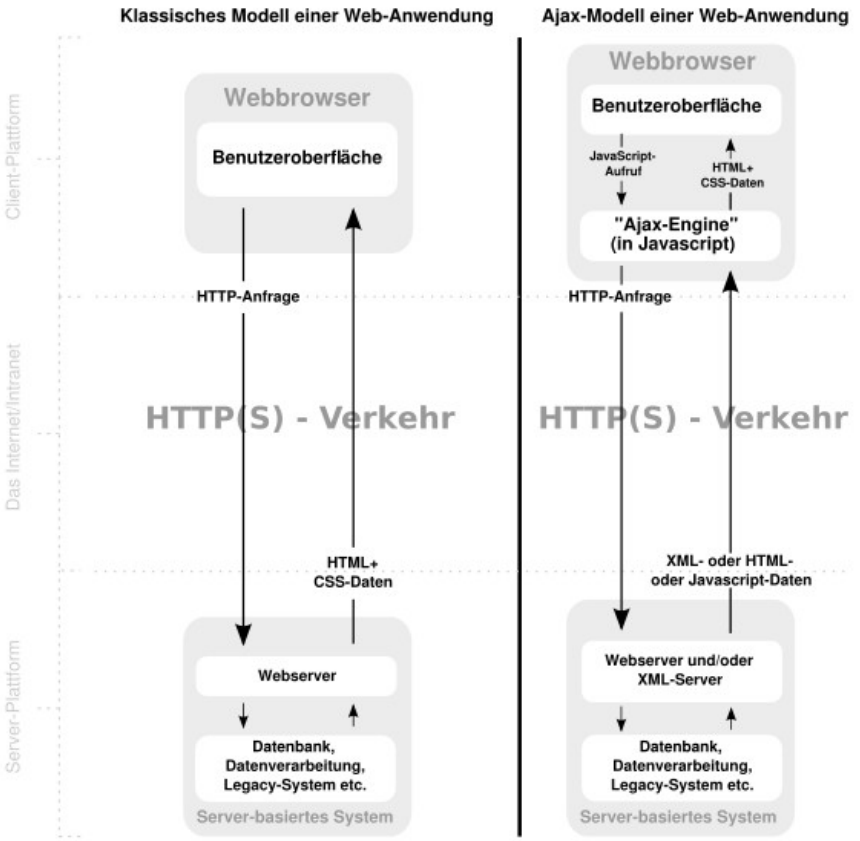


Abb. 12.1: Prinzip der Funktionsweise von AJAX. Grafik Copyright DanielSHaischt

## 12.3 Das XMLHttpRequest-Objekt

| Methode | Erklärung |
|---|---|
| open(method, url, async, user, password) | Throws a SYNTAX_ERR exception if one of the following is true:<br>• method is not a valid HTTP method.<br>• url cannot be resolved.<br>• url contains the „user:password“ format in the userinfo production.<br>• Throws a SECURITY_ERR exception if method is a case-insensitive match for CONNECT, TRACE or TRACK.<br>• Throws a SECURITY_ERR exception if the origin of url does not match the XMLHttpRequest origin.<br>• Throws a NOT_SUPPORTED_ERR exception if the scheme of url is not supported. |
| setRequestHeader(header, value) | Appends an header to the list of author request headers or if the header is already in the author request headers its value appended to.<br>• Throws an INVALID_STATE_ERR exception if the state is not OPENED or if the send() flag is true.<br>• Throws a SYNTAX_ERR exception if header is not a valid HTTP header field name or if value is not a valid HTTP header field value. |
| send(data) | Initiates the request. The optional argument provides the request entity body. The argument is ignored if request method is GET or HEAD.<br>• Throws an INVALID_STATE_ERR exception if the state is not OPENED or if the send() flag is true. |
| abort() | Cancels any network activity. |
| getResponseHeader(header) | Returns the header field value from the response of which the field name matches header, unless the field name is Set-Cookie or Set-Cookie. |
| getAllResponseHeaders() | Returns all headers from the response, with the exception of those whose field name is Set-Cookie or Set-Cookie2. |

Tab. 12.1: Methoden

| Eigesnchaft | Bedeutung |
|---|---|
| responseText | Returns the text response entity body. |
| responseXML | Returns the document response entity body. |
| status | Returns the HTTP status code. |
| statusText | Returns the HTTP status text. |

Tab. 12.2: Eigenschaften

| Ereignis | Erklärung |
|---|---|
| onreadystatechange | Definiert Callback Methode, die aufgerufen wird, wenn sich der readyState ändert. |

Tab. 12.3: Ereignisse

| Status | Wert | Bedeutung |
|---|---|---|
| UNSENT | 0 | The object has been constructed. |
| OPENED | 1 | The open() method has been successfully invoked. During this state request headers can be set using setRequestHeader() and the request can be made using the send() method. |
| HEADERS_RECEIVED | 2 | All redirects (if any) have been followed and all HTTP headers of the final response have been received. Several response members of the object are now available. |
| LOADING | 3 | The response entity body is being received. |
| DONE | 4 | The data transfer has been completed or something went wrong during the transfer (e.g. infinite redirects). |

Tab. 12.4: readyState

## 12.4 Demo

```
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"
        >
    <title>AJAX Test</title>
    <script type="text/javascript">
    <!--
      var req = null;

      function processRequest() {
        switch(req.readyState) {
          case 4:
            if(req.status == 200) {
              document.getElementById("request").innerHTML = req.
                  responseText;
            }
            else
              alert("Fehler: " + req.statusText);
            break;
          default:
            return false;
            break;
        }
      }

      function initRequest() {
        try {
          req = new XMLHttpRequest();
        }
        catch(e) {
          alert("Requestobjekt konnte nicht erstellt werden");
        }
        req.open("GET", "data.txt", true);
        req.onreadystatechange = processRequest;
        req.setRequestHeader("Content-Type", "application/x-www-form-
            urlencode");
        req.send(null);
      }
    -->
    </script>
  </head>
  <body>
    <h1>AJAX Test</h2>
    <h2>Request senden</h2>
    <p>
      <input type="button" onclick="initRequest();" value="Request
          senden" />
    </p>
```

```
    <h2>Ergebnis Request</h2>
    <p>
      <div id="request"></div>
    </p>
  </body>
</html>
```

### 12.4.1 Erläuterungen

#### Aufbau und Funktionsweise des Demo-Programms

Es gibt ein normales HTMl-Dokument mit Inline JavaScript Code. Im Body des HTML-Dokumentes befindet sich eine Schaltfläche zum Absetzen des AJAX Requests und ein div-Container für die Antwort auf den AJAX Request. Die Schaltfläche ruft den JavaScript Code *initRequest()* auf. In dieser JavaScript Funktion wird das Request Objekt erzeugt. Dem Eventhandler onreadystatechange wird die Callback Funktion *processRequest()* zugewiesen. Diese Funktion reagiert auf die Antwort des Servers und fügt die Antwort in dem div-Container ein.

#### Erzeugen des AJAX Request Objekts

Hier

```
req = new XMLHttpRequest();
```

wird das Request Obket erzeugt. Schlägt dies fehl, wird eine Exception ausgelöst. Der einfachheithalber wurde auf die Kompatibilität zu anderen und älteren Browser verzichtet. Der Internet Explorer unterstütz das XMLHttpRequest Objekt zum Beispiel erst ab der Version 7. Davor musste man das ActiveXObjekt *Msxml2.XMLHTTP* oder *Microsoft.XMLHTTP* benutzen:

```
req = new ActiveXObject(Msxml2.XMLHTTP
```

oder

```
req = new ActiveXObject(Microsoft.XMLHTTP
```

### Auslösen eines asynchronen Requests

```
req.open("GET", "data.txt", true);
req.onreadystatechange = processRequest;
req.setRequestHeader("Content-Type", "application/x-www-form-urlencode
    ");
req.send(null);
```

Mit der Methode *open* wird der Request erzeugt. Der erste Parameter bestimmt die Art des Requests, ob POST oder GET. Der zweite Parameter ist die URL an der der Request gesendet werden soll. Dies kann eine Text-, JSON- oder XML-Datei sein, welche Daten liefert oder das Ergebnis eines PHP Scriptes. Der letzte Parameter gibt an, ob der Aufruf synchron (false) oder asynchron (true) erfolgen soll.

### Definition der Callback Funktion

Mit dem Eventhandler *readystatechange* wird eine Callback Funktion *processRequest()* verknüpft, die immer dann aufgerufen wird, wenn sich der Status des Requests ändert. Siehe dazu die Tabelle 12.4. In dieser Funktion findet eine Fallunterscheidung für den *readyState* statt und es wird der Status geprüft, ob die Anforderung erfolgreich war. Ist der *readyState* DONE, also der Request abgeschlossen, wird geprüft, ob er erfolgreich war (HTTP Errorcode 200) und dann die Antwort des Requests in den dafür vorgesehenen div-Container eingefügt:

```
switch(req.readyState) {
case 4:
  if(req.status == 200) {
    document.getElementById("request").innerHTML = req.responseText;
```

# Literaturverzeichnis

[1] Ralph Steyer: *HTML 4*. Data Becker, 1. Auflage, 1999, 3-8158-1598-3

[2] Internet: *SelfHTML*. http://de.selfhtml.org, Stand: Dezember 2010

[3] Internet: *Wikipedia*. http://de.wikipedia.org/wiki, Stand: März 2011