
Angie Radtke
JOOMLA!
TEMPLATES ENTWICKELN
Barrierefreie & attraktive Designs
von Konzept bis Umsetzung
ADDISON-WESLEY
MUSIC* STORE
Font size Bigger Reset Smaller
Search...
HOME
SAMPLE SITES
JOOMLA.ORG
Time for YOUR music
CD's, DVD's BLUE RAY
You are here: Home
Relax and enjoy your favorite music!
In our store you will find all genres of music at a great price. R&B, HipHop, Soul, Folk, Jazz, Classical, Rhythm and Blues ...
discover all we have to offer
News
ADMINISTRATOR COMPONENTS
All components also are used in the administrator area of your website. In addition to the ones list...
About Joomla!
Getting Started
Using Joomla!
The Joomla! Project
The Joomla! Community
ARCHIVE MODULE
This module shows a list of the calendar months containing archived articles. After you have

3 CSS und HTML – Das Grundgerüst in Form bringen

HTML und CSS gehören zum Grundwissen eines jeden Webdesigners. HTML ist relativ einfach zu erlernen, bis man aber die Tipps und Tricks des CSS beherrscht, braucht es einiges an Übung. Hier ist weniger die Theorie, sondern vielmehr die Praxis das Problem. Die unterschiedlichen Browser setzen die CSS-Anweisungen immer noch unterschiedlich um und können Webdesigner manchmal wirklich zur Weißglut bringen.

Im Rahmen dieses Buches ist es mir nicht möglich, auf die korrekte Verwendung von CSS im Detail einzugehen. Obwohl ich mir der Unvollständigkeit bewusst bin, möchte ich dennoch versuchen, eine kleine Einführung zu Papier zu bringen. Denn die Erfahrungen im Forum haben mir gezeigt, dass auch Menschen mit keinerlei Erfahrung in diesem Themenbereich Websites mit Joomla! erstellen. So zum Beispiel der Vorsitzende des Fußballclubs oder ein engagiertes Mitglied der Kirchengemeinde im Ort.

Ich werde mich bemühen, mich hierbei immer an klassischen Joomla!-Beispielen zu orientieren. Sollten Sie CSS- und HTML-Experte sein, dann können Sie dieses Kapitel also getrost überlesen.

3.1 Ein wenig zur Geschichte

Das Netz spricht HTML. HTML ist eine Seitenbeschreibungssprache und so alt wie das Netz selbst. Tim Berners Lee entwickelte diese Sprache im Jahre 1990. Ihm ging es damals nicht um die Gestaltung von Information, sondern einfach um die rein inhaltliche Struktur. Aus diesem Grund lassen sich mit HTML auch nur semantische Gliederungen abbilden. Dokumenten kann man Überschriftenhierarchien zuweisen, es können Listen erstellt werden, aber auch tabellarische Daten lassen sich in HTML verarbeiten. Gestaltungsmöglichkeiten gab es damals fast keine.

Die Informationen, die man vor vielen Jahren im Web finden konnte, waren für viele Menschen weitaus zugänglicher als so manches Dokument heute.

Erst mit dem Wunsch, Inhalte auch optisch anpassen zu können, kam es in vielen Bereichen zu Zugangsbeschränkungen für Menschen mit Handicap.

Tabellen, die eigentlich zur Präsentation von Datenstrukturen gedacht sind, wurden missbraucht, um Inhalte am Bildschirm zu platzieren. Auf die semantische Auszeichnung von Überschriften wurde weitgehend verzichtet, weil sich Tabellenzeilen optisch einfacher gestalten ließen.

CSS ist eine Art Formatierungssprache für HTML. Als ich 1999 mit dem Webdesignen begann, war eines der ersten Bücher zum Thema, das ich las, *Cascading Style Sheets* von Eric Meyer. Ich arbeitete mich ein und musste schon nach einigen Wochen feststellen,

dass die Zeit für den Einsatz von CSS wegen der fehlenden Browserunterstützung noch nicht reif war. Und so lernte ich das Layouten mit Tabellen.

Heute sieht das zum Glück ganz anders aus. Durch den Einsatz von CSS steht der Trennung von Content und Layout nichts mehr im Wege.

Um standardkonforme und performante Seiten zu entwickeln, ist der Einsatz von CSS unumgänglich und so ist ausreichendes Wissen zu diesem Thema auch für die Gestaltung von Joomla!-Seiten notwendig.

Zu den Themen HTML und CSS gibt es einige sehr gute Bücher. Ich kann hier nur einen groben Überblick geben, was HTML und CSS ist und wie man beides sinnvoll einsetzen kann:

- *Einführung in XHTML, CSS und Webdesign* von Michael Jendryschik. Addison-Wesley 2008, ISBN 978-3-8273-2739-0.
- *Webpublishing mit HTML und CSS* von Laura Lemay und Rafe Colburn. Addison-Wesley 2011. ISBN 978-3-8273-3061-1
- *Das große Little Boxes-Buch* von Peter Müller. Markt+Technik 2011, ISBN 978-3-8272-4714-8

3.2 Welche Version von HTML sollte ich verwenden?

Eine Frage, die in der Praxis oft zu Unsicherheiten führt: Welche Version der Markupsprache (X)HTML setze ich ein? Zur Auswahl stehen lediglich HTML 4.01 und XHTML 1.0 – jeweils in den Versionen Transitional, Strict und Frameset. Neuerdings auch HTML5.

Der Unterschied zwischen HTML 4.01 und XHTML 1.0 wird oft überschätzt – im Hinblick auf Sprachumfang und Leistungsfähigkeit stimmen beide genau überein und unterscheiden sich nur in einigen Formalien. Eine automatische Konvertierung zwischen HTML 4.01 strict und XHTML 1.0 strict ist jederzeit möglich.

Die HTML-Ausgabe von Joomla! selbst basiert auf XHTML, so dass sich die Frage nach der Version erübrigt.

Bedeutender ist für beide Sprachen der Unterschied zwischen der *Strict*- und der *Transitional*-Version. *Transitional* erlaubt die Verwendung von nicht mehr empfohlenen oder missbilligten Attributen.

Sicherlich hat der ein oder andere von Ihnen schon einmal den Ausdruck *das ist doch deprecated!* gehört. *Deprecated* bedeutet „nicht mehr erwünscht“. Zu diesen nicht mehr erwünschten Attributen gehören zum Beispiel das `vspace`- und `hspace`-Attribut des Image-Elementes oder das Attribut `align`.

Im Zusammenhang mit Joomla! möchte ich Ihnen dennoch zur Wahl von Transitional raten, auch wenn die Standardausgabe von Joomla! Strict erlaubt:

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

Das wundert Sie sicherlich, denn Strict wäre die standardkonformere und elegantere Wahl. Der Grund liegt in den häufig verwendeten Texteditoren, die bei Joomla! mitgeliefert werden.

Diese Editoren nutzen diese veralteten Attribute leider immer noch. Der HTML-Validator, ein Tool, das Ihre Seite auf Korrektheit prüft, würde zwangsläufig diese „falsch verwendeten" Attribute bemängeln.

Sollten Sie jedoch auf die Editoren verzichten, den Code mit der Hand editieren oder den Editor *Xstandard* nutzen, ist es sicherlich besser, die Version Strict zu wählen:

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

HTML5

Das Web bewegt sich in großen Schritten auf HTML5 zu. Schon jetzt findet man eine Vielzahl von HTML5-Seiten im Netz.

Stöbern Sie einmal selbst. Fündig werden Sie unter `http://html5gallery.com/`. Selbst große Unternehmen wie e-plus oder Google setzen schon jetzt auf HTML5.

Begeistert hat mich besonders die Deklaration des Dokumententyps:

```
<!Doctype html>
```

Endlich ist etwas einmal einfach!

Auch die semantisch bedeutungsvollen Elemente sind besonders interessant. Hier fallen mir sofort Elemente wie `nav` für die Navigation, `header`, `footer`, `aside` für Zusatzinformation, `section` für Bereiche und `article` für Inhalte auf. Allein diese Elemente stellen eine große Bereicherung dar.

Während Opera, Firefox, Safari (jeweils in der aktuellen Version) und der Internet Explorer 9 wunderbar mit den Elementen umgehen können, stellen sich der Internet Explorer 6, 7 und 8 quer. Die Elemente lassen sich nicht per CSS ansprechen und stylen – sie sind diesen Browsern unbekannt. Zur Lösung dieses Problems gibt es aktuell zwei Möglichkeiten. Entweder fügen Sie um diese Elemente althergebrachte `div`-Elemente ein und stylen sie entsprechend oder Sie fügen per JavaScript die unbekannten Elemente ins DOM des Browsers hinzu.

```
<!--[if lt IE 9]>
<script type="text/javascript">
document.createElement('section');
document.createElement('nav');
document.createElement('article');
document.createElement('header');
document.createElement('footer');
document.createElement('aside');
document.createElement('figure');
document.createElement('legend');
```

```
</script>
<![endif]-->
```

Der Nachteil dieser Methode liegt darin, dass sie bei deaktiviertem JavaScript natürlich nicht funktioniert.

Innerhalb des beez5-Templates können Sie zwischen XHTML und HTML5 als darzustellende Markupsprache im Backend wählen. Realisiert wurde diese Möglichkeit mit Hilfe der so genannten Template-Overrides, deren Funktionsweise ich in einem späteren Kapitel eingehender erläutere.

3.3 Das HTML-Grundgerüst

Das HTML-Grundgerüst Ihrer Seite bestimmen Sie in Ihrem Template selbst. Es steuert den generellen Aufbau der Seite und legt fest, wo sich innerhalb des Dokumentenflusses die einzelnen Seitenbereiche wie Header, Footer etc. befinden. Dieses Gerüst lässt sich mit CSS wunderbar in Form bringen. Joomla! selbst versieht nur dynamische Inhalte mit vorstrukturiertem Markup.

Dieses Markup hat sich in der Version 1.6 deutlich verändert, wurde modernisiert und auf den neuesten Stand der Technik gebracht. Aber dazu und zum Aufbau des Gerüsts an späterer Stelle mehr.

3.4 Eine kurze Einführung in CSS

3.4.1 Einleitung

CSS in einem Kapitel? Eigentlich ein Ding der Unmöglichkeit. Zu diesem Thema gibt es eigene Bücher, sogar ganze Buchreihen. Mir wird es nicht gelingen, in einem Kapitel oder sogar nur einem Kapitelabschnitt die gesamte Funktionsweise von CSS zu erläutern. Aber ich kann Ihnen ein Grundverständnis für die Möglichkeiten von CSS geben, und das möchte ich im Folgenden versuchen.

CSS ist eine Auszeichnungssprache für strukturierte Dokumente. Mit dem Einsatz von CSS haben Sie direkten Einfluss auf die Gestaltungen des ausgegebenen HTML-Codes. Sie können Überschriften, Listen und Absätze formatieren oder Inhalte strukturiert auf dem Bildschirm anordnen, was sicher die hohe Schule der CSS-Nutzung darstellt.

Auf `http://edition-w3c.de/TR/1998/REC-CSS2-19980512/` finden Sie eine komplette CSS- Referenz.

3.5 Einbinden der CSS-Anweisungen

Es gibt die unterschiedlichsten Wege, um eine CSS-Anweisung in Ihre Seite einzubinden. Ihre Gewichtung innerhalb aller CSS-Deklarationen hängt von ihrer Positionierung innerhalb eines Dokuments ab. Von dieser Funktionalität ist auch der Name „Cascading Stylesheets“ abgeleitet.

3.5.1 Externe Einbindung als Datei oder direkt im Dokumentenkopf

Zum einen können Sie Style-Angaben in einer externen Datei positionieren:

```
<link rel="stylesheet" href="/templates/beez/css/layout.css"
type="text/css" media="screen,projection" title="" />
```

Dies geschieht im Kopf der Seite. Diese Wahl wird sicher am häufigsten getroffen und sie ist sicher auch die effektivste, da man so Inhalt und Darstellung völlig voneinander trennen kann. Wenn Sie alle Style-Angaben in eine solche Datei auslagern, können Sie bei einem Relaunch Ihrer Seite das Design ändern, ohne dass Sie den Code selbst anfassen müssen.

Sie können beliebig viele CSS-Dateien auf dem oben genannten Weg einbinden. Das macht zum Beispiel Sinn, wenn Sie die Positionierung der Inhalte von der Farbgebung trennen möchten. Es lassen sich auch spezielle Style-Dateien für den Internet Explorer 6 einbinden oder gezielte Dateien für die Druckansicht gestalten. Wenn Sie sich die `index.php` von Beez anschauen, sehen Sie dort eine ganze Reihe von CSS-Dateien.

Desweiteren können Sie Style-Angaben in den Kopf der Seite direkt einbinden. Viele frei verfügbare Joomla!-Erweiterungen nutzen diese Methode noch, um ihre speziellen Stylesheets zu integrieren.

```
<style type="text/css">
<!--
div.mycomponentdiv
{
background:#000}
}
-->
</style>
```

Diese Style-Angaben der externen Joomla!-Erweiterungen passen häufig nicht zu dem eigenen Layout und man würde gerne darauf verzichten. Da sie aber inhaltlich nach unserem eigenen CSS, das wir mit `link rel` eingebunden haben, im Quelltext erscheinen, haben sie eine höhere Priorität: Die Kaskade kommt zum Einsatz.

Würde man diese Styles jedoch vor unserem mit `link rel` eingefügtem CSS platzieren, gäbe es das Problem nicht.

Schlimmer noch ist es, wenn externe Komponenten Inline-Styles verwenden. Unsere Chance, diese zu verändern, ist gleich null, denn Inline-Styles haben die höchste Priorität und überschreiben alles andere.

3.5.2 Inline-Styles

HTML-Elemente verfügen über das Attribut `style`, über das sie sich individuell formatieren lassen:

```
<p style="color:#cc0000"> Das ist ein Absatz mit roter
Schriftfarbe.</p>
```

Auf den Einsatz von Inline-Styles sollten Sie aus unterschiedlichsten Gründen verzichten:

- Bei der Neugestaltung einer Website müssen Sie die Inline-Styles mühevoll aus dem bestehenden Code entfernen.
- Menschen, die an Sehbehinderungen leiden, erstellen sich individuelle Styles in ihren Browsern und wenden diese auf alle Websites an. Da Inline-Styles die höchste Priorität haben, lassen sich diese Styles auf eine solche Seite nicht anwenden.
- Inline-Styles blähen den HTML-Code unnötig auf.

Aber trotz alledem – lassen Sie sich nicht von all jenen beeindrucken, die Inline-Styles verteufeln. Es gibt durchaus Situationen, wo sie ganz praktisch sein können. Manchmal möchte man zum Beispiel innerhalb eines Moduls ein Hintergrundbild einfügen und dieses vielleicht öfter austauschen. Hier macht es wenig Sinn, die CSS-Dateien zu adaptieren. Das wäre ein weitaus größerer Aufwand, als Inline-Styles zu verwenden.

Fazit: Die Styles sind am besten in einer externen Datei im Kopf der Seite aufgehoben. Bei der zu verwendenden Datei handelt es sich um eine reine Textdatei, die Sie mit einem ganz einfachen Editor wie dem Notepad bearbeiten können.

Ich selbst gehöre noch zur altmodischen Sorte und bearbeite alle meine Dateien mit Ulli Meybohms *Phase 5*. ;-)

3.5.3 Selektoren CSS

Selektoren – das klingt erst einmal verwirrend und kompliziert. Aber keine Sorge, das ist es gar nicht.

Das CSS muss wissen, an welcher Stelle es gelten soll und dazu braucht es Selektoren. HTML-Elemente wie `p` `h1-h6`, `div` und `span` sind solche Selektoren. Man kann den Elementen allerdings auch Klassen und IDs zuweisen, die dann als Selektoren fungieren, doch dazu später mehr.

3.5.4 Elementselektoren

Zunächst einmal erläutere ich Ihnen die Syntax anhand einer ganz einfachen CSS-Anweisung.

Im obigen Beispiel mit den Inline-Styles haben wir einen Absatz mit roter Schriftfarbe versehen. Diese Anweisung können Sie jetzt auf alle `p`-Elemente innerhalb eines Dokumentes anwenden:

```
p {color:#cc0000;}
```

`p` fungiert hier als Selektor, darauf folgen geschweifte Klammern, in denen die Eigenschaft steht – durch einen Doppelpunkt von ihrem Wert getrennt. Ein Semikolon schließt die Anweisung ab. (In diesem Falle könnten wir das Semikolon auch weglassen, da dies die letzte – oder besser gesagt – die einzige Anweisung in unserer Deklaration ist.)

Das Semikolon ist notwendig, wenn wir einem Element mehrere Deklarationen mitgeben möchten:

```
p
{
 color:#cc0000;
 background:#eee
 }
```

Möchten Sie diese Anweisung unterschiedlichen Elementen zuweisen, können Sie sie auch gruppieren, indem Sie die Elemente durch ein Komma voneinander trennen. Aber Achtung: Auf den letzten Selektor folgt **kein** Komma!

```
p, h1, h2
{
 color:#cc0000;
 background:#eee
 }
```

3.5.5 Mehrmals verwendbar: CSS Klassen

Möchten Sie nur bestimmten Absätzen in einem Dokument, die Schriftfarbe Rot zuweisen, ist dies mit der Vergabe einer Klasse am einfachsten zu lösen.

Der Einsatz einer Klasse macht aber nur dann Sinn, wenn Sie diese tatsächlich mehrmals nutzen möchten.

In Ihrem Quelltext sieht das dann zum Beispiel so aus:

```
<p class="attention"> Absatz mit roter Schrift.</p>
```

Hier das entsprechende CSS:

```
.attention  {color:#cc0000;}
```

Sie sehen, dass sich vor dem Wort `attention` ein Punkt befindet. Dieser Punkt verweist innerhalb des CSS auf eine Klasse.

Klassen lassen sich auf verschiedenste HTML-Elemente anwenden. So würde auch bei

```
<h1 class="attention"> Das ist eine Überschrift </h1>
```

der Überschrift der ersten Ordnung die Schriftfarbe Rot zugewiesen.

Gruppieren von Klassen

Das Gruppieren von Klassen ist ebenfalls möglich und sogar besonders flexibel. Ich möchte dies schon jetzt am Beispiel der Joomla!-Standardausgabe erklären, da dies weitaus mehr Praxisbezug hat, als wenn ich mich auf theoretischer Ebene bewege. Da Sie im Moment aber noch nichts über den Aufbau der Joomla!-Templates wissen, bitte ich Sie: Lassen Sie sich jetzt nicht verwirren! Lesen Sie zunächst einmal hier weiter und laufen Sie nicht gleich zu Ihrem Rechner, um sich das Template anzusehen.

Wie Sie sicherlich wissen, haben Sie mit Joomla! die Möglichkeit, über die Parameter in den Blogansichten, die Anzahl der darzustellenden Spalten festzulegen. In der Tabellenvariante von Joomla! 1.5 ist das kein Problem: Die Tabelle wird einfach um eine weitere Spalte erweitert, wenn man die Werte in den Parametern um eins erhöht. Nichts verrutscht, Ihre Seite wird einfach entsprechend aufgezogen.

Jetzt realisieren wir diese Darstellung mittels einzelner `div`-Elemente. `div`s sind semantisch bedeutungsleere Elemente, die sich auf dem Bildschirm anordnen lassen, um so das visuelle Raster einer Seite aufbauen zu können. Sie können beliebig viele andere Elemente enthalten und bedenkenlos verschachtelt werden:

```
<div><p>Absatz</p></div>
```

Bei der mehrspaltigen Ansicht brauchen die Elemente eine gewisse Breite, um richtig angeordnet zu werden. Das bedeutet, Sie brauchen zumindest eine Klasse, der Sie eine Breite mitgeben können. Möchten Sie jetzt die erste Spalte farblich anders gestalten als die zweite, benötigen Sie eine weitere Klasse. Sie haben nun die Möglichkeit, einem Bereich zwei Klassen zuzuweisen.

Die Breite lässt sich über die Eigenschaft `width` festlegen und erwartet einen absoluten Wert in Pixeln oder einen relativen Wert in `em` oder `%`. Mehr dazu erfahren Sie im Abschnitt *3.7, Positionierung und Boxmodell*.

```
<div class="Breite Farbe"> <p> Absatz </p> </div>
<div class="Breite Farbe2"> <p> Absatz </p> </div>
```

Das entsprechende CSS sähe wie folgt aus:

```
.Breite {width:200px}
.Farbe { background:#cccc99} // hellgrün
.Farbe2 {background:#999900} // grün
```

Abbildung 3.1
So sieht das Ergebnis aus

Bei einspaltiger Darstellung haben Sie folgenden Code:

```
<div class="item column-1" ></div>
```

Bei zweispaltiger Darstellung sieht der Code so aus:

```
<div class="item column-1" ></div>
<div class="item column-2" ></div>
```

Und bei dreispaltiger Darstellung so:

```
<div class="item column-1" ></div>
<div class="item column-2" ></div>
<div class="item column-3" ></div>
```

Sie hätten nun die Möglichkeit, der Klasse `item` generelle Formatierungen mitzugeben, die für alle Spalten gelten sollen. Das könnte zum Beispiel die Schriftart sein. Mit den Klassen `column-1`, `column-2` und `column-3` ließen sich die Breite und die Positionierung der unterschiedlichen Spalten festlegen.

Das Ganze könnte dann so aussehen:

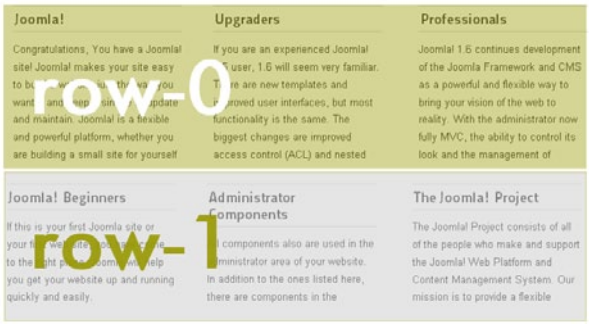
Abbildung 3.2
Mittels CSS positionierte und gestaltete Spalten.

Zusätzlich können Sie auch jede Zeile separat gestalten.

Nehmen wir an, wir hätten sechs Artikel und drei Spalten, dann würden diese in zwei Zeilen dargestellt. Der Joomla!-Code sähe dann folgendermaßen aus.

```
<div class="items-row cols-3 row-0">
   <div class="item column-1" ></div>
   <div class="item column-2" ></div>
   <div class="item column-3" ></div>
</div>
<div class="items-row cols-3 row-1">
   <div class="item column-1" ></div>
   <div class="item column-2" ></div>
   <div class="item column-3" ></div>
</div>
```

ABBILDUNG 3.3
Spalten und Zeilen



Joomla! umschließt die Artikel innerhalb einer Zeile immer mit einem eigenen `div`-Element. Dieses Element besitzt selbst wiederum seine eigenen Klassen. Eine universelle Klasse, die bei jeder Zeile gleich ist, `items-row`, eine weitere Klasse, die Auskunft über die Anzahl der Spalten gibt sowie eine, die sagt, welche Zeile gerade ausgegeben wird:

- `.items-row` universelle Klasse, die bei jeder Zeile gleich ist.
- `.cols-3` gibt Auskunft über die Anzahl der Spalten.
- `.row-0 -row-xx` gibt Auskunft über die Anzahl der Zeilen.

Daraus ergibt sich eine Vielzahl von Gestaltungsmöglichkeiten.

Je nach gewählter Spaltendarstellung sind Sie in der Lage, ganz individuelle Formatierungen vorzunehmen. Der Fantasie sind dabei keine Grenzen gesetzt.

Ein weiterer großer Vorteil: Sie sind nicht mehr auf die Anzahl der darstellbaren Spalten beschränkt. Sie können gefahrlos zehn Spalten im Backend auswählen, müssen diese allerdings dann mittels CSS entsprechend formatieren.

3.5.6 ID-Selektoren

Das Wichtigste zuerst: IDs dürfen innerhalb einer Datei nur ein einziges Mal vorkommen. Das erklärt auch, warum sie häufig zum Positionieren einzelner Seitenbereiche im HTML-Grundgerüst eingesetzt werden.

Im HTML-Code sieht das folgendermaßen aus:

```
<div id=" header">
<p> Alles, was im Kopf ist</p>
</div>
```

Innerhalb der CSS-Datei können Sie wie folgt darauf zugreifen:

```
#header
{
background:#000;
color:#fff
}
```

Die # ist hier im CSS der Indikator für die IDs.

Die Positionierung einzelner Seitenbereiche ist sicherlich die hohe Schule des CSS und manchmal wegen der unterschiedlichen Browser nicht immer ganz einfach zu realisieren.

3.5.7 Kontextselektoren

Die Kontextselektoren sind eine sehr praktische und Code sparende Methode, um einzelne Seitenbereiche mittels CSS zu formatieren.

Nehmen wir an, wir hätten innerhalb unseres Headerbereiches eine Liste und wollten diese formatieren.

Der HTML-Code sähe wie folgt aus:

```
<div id="header">
   <ul>
      <li> Listenpunkt 1 </li>
      <li> Listenpunkt 2 </li>
   </ul>
</div>
```

Wir könnten jetzt so darauf zugreifen:

```
#header ul li {color:#cc0000;}
```

Das bedeutet, alle Listenpunkte innerhalb des Headers bekämen die Schriftfarbe Rot. Elegant, oder?

3.5.8 Kindselektoren

Die im nächsten Absatz beschriebenen Selektoren sind sehr praktisch, aber leider werden sie vom Internet Explorer 6 nicht unterstützt.

```
body >p
```

formatiert alle `p`, die ein direktes Kind von `body` sind.

Direktes Kind bedeutet hier, dass kein weiteres Element dazwischen liegt. Im nachfolgenden Beispiel würde der Selektor keine Anwendung finden, da der Absatz kein direktes Kind des `body`-Elementes, sondern des untergeordneten `div`s ist.

```
...
<body>
<div>
<p> Das ist ein Absatz in einem div </p>
</div>
</body>

ul li:firstchild
```

formatiert das erste `li`-Element innerhalb einer Liste.

```
p + div
```

formatiert das erste `div`, das auf einen Absatz folgt.

Solange Sie ein separates Stylesheet für den Internet Explorer 6 erstellen, indem Sie die fehlende Implementierung durch eine andere Lösung ausgleichen, ist gegen die Verwendung dieser Selektoren nichts einzuwenden – sie ist sogar zu begrüßen.

3.5.9 Universalselektoren

Sicherlich kennen Sie die Wildcard * von der Windows Suche. Wenn Sie eine Datei mit der Endung `png` finden wollen, geben Sie einfach `*.png` ein und Windows durchsucht Ihren Rechner nach allen Dateien mit dieser Endung. Dabei ist es unerheblich, wie die Datei davor korrekt bezeichnet ist. Wildcards sind Platzhalter für andere Zeichen.

Innerhalb von CSS wird diese Verwendung übernommen.

```
* {margin:0; padding:0}
```

setzt zum Beispiel die Innen- und Außenabstände aller Elemente auf 0.

Da die unterschiedlichsten Browser automatisch verschiedensten HTML-Elementen Abstände zuweisen, ist dies eine sinnvolle Formatierung, um alle Browser zu Beginn auf den gleichen Stand zu bringen.

```
body * p {margin:0; padding}
```

setzt zum Beispiel die Abstände eines Absatzes auf 0, wenn zwischen ihm und dem `body` noch ein anderes Element vorkommt.

3.5.10 Pseudoklassen-Selektoren

Eine besondere Form der Formatierung ist die über die so genannten Pseudoklassen. Pseudoklassen nehmen keinen Einfluss auf HTML-Elemente an sich, sondern überschreiben Browser-interne Formatierungen.

Die wichtigste Pseudo-Klasse ist sicherlich die zur Formatierung von Links. Wenn Sie sich schon einmal durch die Konfiguration Ihres Browsers geklickt haben sollten, ist Ihnen vielleicht aufgefallen, dass Sie die Möglichkeit haben, zum Beispiel die Farbe für Links festzulegen. In den meisten Fällen können Sie dort auswählen, was Sie wollen, es wird keine Auswirkung mehr auf die Darstellung moderner Webseiten haben, die Sie ansurfen. Denn diese nutzen CSS, um Ihre Voreinstellung zu überschreiben.

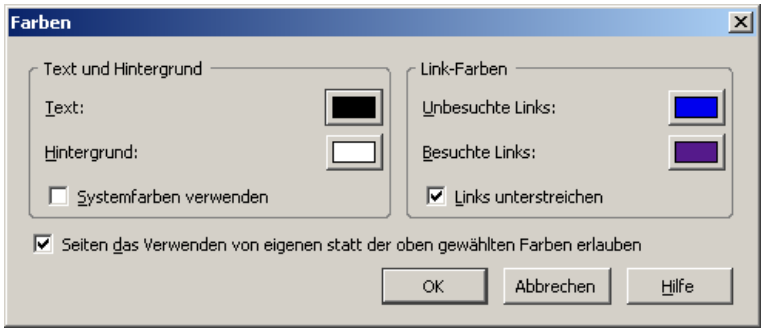


Abbildung 3.4
So sieht das zum Beispiel im Firefox aus.

- `a:link` für einen normalen Link
- `a:visited` für einen besuchten Link
- `a:focus` für einen Link, der beim Tabben durch die Seite aktiv wird
- `a:hover` für den Zustand des Links beim Überfahren mit der Maus
- `a:active` für einen gerade aktiven Link/Internet Explorer 6 beim Tabben aktiv

Für die Formatierung ist die Reihenfolge Ihrer Anordnung innerhalb des CSS wichtig, da es sonst zu Darstellungsfehlern kommt.

Weitere Pseudoklassen oder Pseudoelemente sind zum Beispiel

- `li:first-child` formatiert das erste Kindelement des Listenpunkts
- `p:first-letter` formatiert den ersten Buchstaben eines Absatzes
- `p:first-line` formatiert die erste Zeile eines Absatzes

Es gibt noch weitaus mehr Pseudoklassen, als die hier erwähnten. Viele davon werden leider vom Internet Explorer 6 nicht unterstützt.

3.5.11 Attribut-Selektoren

Nur der Vollständigkeit halber: Mit Hilfe der Attribut-Selektoren lassen sich Attribute von HTML-Elementen, wie zum Beispiel das `title`-Attribut einer Überschrift formatieren.

```
h1[title] { color: 0000ff; }
```

Diese Selektoren werden noch selten verwendet. Der Grund liegt unter anderem darin, dass es keine flächendeckende Browserunterstützung gibt.

Weiterführender Link zum Thema:
`http://edition-w3c.de/TR/1998/REC-CSS2-19980512/kap05.html`

3.6 Vererbung

Bestimmte CSS-Eigenschaften werden von Elternelementen an ihre Kinder vererbt. Das System der Vererbung erspart so manche Schreiberei.

Hat das `body`-Element zum Beispiel die Schriftfarbe Schwarz, werden auch alle Absätze darin diese Schriftfarbe annehmen, wenn nicht explizit etwas anderes angegeben wird.

Es werden nicht alle Eigenschaften vererbt – bei `margin` oder `border` zum Beispiel ist das nicht der Fall.

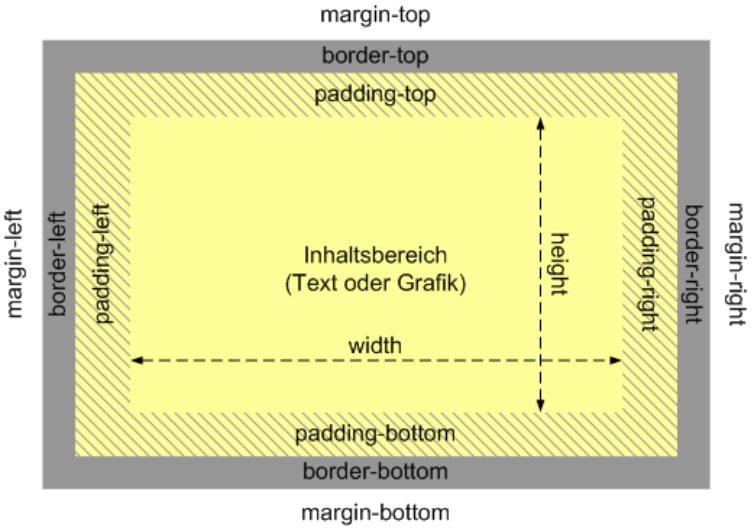
3.7 Positionierung und Boxmodell

Die Positionierung einzelner Elemente am Bildschirm ist sicherlich die schwierigste Aufgabe, die man mit CSS erledigen kann.

Man muss sich vorstellen, dass alle Elemente, die in einer Seite vorkommen, rechteckige Boxen oder Kästen auf dem Bildschirm einnehmen, um so den Platz für ihren Inhalt zu reservieren.

Jeder dieser Boxen lassen sich Eigenschaften über deren Breite, ihren Abstand nach außen und innen und ihre Umrandung mitgeben. Man kann ihnen auch eine Höhe zuweisen, was aber in der Regel wegen dynamischen Inhalts etwas tricky ist. Desweiteren lässt sich auch ihre visuelle Position auf dem Bildschirm festlegen.

Abbildung 3.5
Schematische Darstellung des Boxmodells



Am verständlichsten wird das Ganze sicherlich durch ein Beispiel:

```
<div id="inhalt">
<h1> Das ist eine Überschrift</h1>
<p> Wir möchten jetzt diese Box formatieren. </p>
</div>
```

Wir haben dem Container die `id="inhalt"` zugewiesen und können ihn jetzt über den `id`-Selektor formatieren.

Zuerst weisen wir der Box einen Rahmen zu, um deutlicher erkennen zu können, was wir tatsächlich tun.

```
#inhalt { border:solid 1px #0000cc}
```

Abbildung 3.6
Das Ergebnis sieht wie folgt aus

Man sieht, dass der Inhalt direkt am Rahmen klebt, was sehr unschön aussieht. Aus diesem Grund weisen wir der Box einen Innenabstand zu. Innenabstände lassen sich über die Eigenschaft padding formatieren. Man kann sie einzeln für jeden Seitenbereich der Box festlegen.

- `padding-top`: *Innenabstand von oben*
- `padding-right`: *Innenabstand von rechts*
- `padding-bottom`: *Innenabstand von unten*
- `padding-left`: *Innenabstand von links*

Damit man nicht alle Eigenschaften einzeln aufführen muss, lassen sie sich auch gruppieren. Dabei beginnen wir mit `padding-top` und legen dann im Uhrzeigersinn die anderen Seitenabstände fest.

```
#inhalt
{
border:solid 1px #0000cc;
padding:10px 20px 10px 20px;
}
```

Abbildung 3.7
Box mit Padding

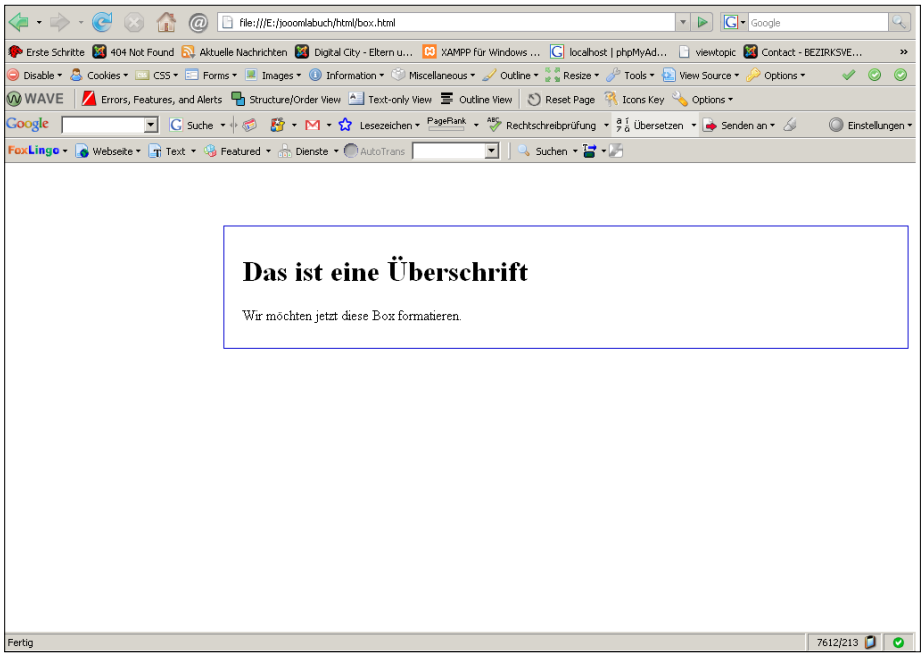
Desweiteren lässt sich auch der Außenabstand einer Box festlegen.

- `margin-top`: *Außenabstand oben*
- `margin-right`: *Außenabstand rechts*
- `margin-bottom`: *Außenabstand unten*
- `margin-left`: *Außenabstand links*

```
#inhalt
{
border:solid 1px #0000cc;
padding:10px 20px 10px 20px;
margin: 70px 0px 0px 240px
}
```

ABBILDUNG 3.8

Das Ergebnis jetzt mit Browserleiste, damit man deutlicher sieht, dass unser Container nun um 240px von linken Rand und 70px vom oberen Rand weggerückt ist.



Um diesen Container im zweiten Schritt in ein wirkliches Layout zu integrieren, können wir ihm eine Breite zuweisen.

In den seltensten Fällen gibt es Seiten mit nur einem Element. Um unterschiedlichste Designs umsetzen zu können, muss ein Grundverständnis für die CSS-Positionierungsmechanismen vorhanden sein.

Im folgendem Beispiel geht es um die Umsetzung eines einfachen zweispaltigen Layouts mit Hilfe der CSS-Eigenschaften `float` und `margin`. Dieses Beispiel dient lediglich dazu, Ihnen ein Gefühl für die CSS-Positionierungsmethoden zu geben. Die gezeigte Vorgehensweise ist nicht immer das Mittel der Wahl, um Elemente nebeneinander am Bildschirm zu positionieren. Die Eigenschaften des Containers `id="inhalt"` bleiben erst einmal identisch.

Der Quellcode des Beispiels hat folgende Struktur:

```
<div id="navigation">
   <ul>
   <li><a href="#">Menulink</a></li>
   <li><a href="#"> Menulink </a></li>
   <li><a href="#"> Menulink </a></li>
   <li><a href="#"> Menulink </a></li>
   <ul>
</div>
<div id="inhalt">
   <h1> Das ist eine Überschrift</h1>
   <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
      Aenean commodo ligula eget dolor. Aenean massa. Cum sociis
      natoque penatibus et magnis dis parturient montes, nascetur
      ridiculus mus. Donec quam felis, ultricies nec,
      pellentesque eu, pretium quis, sem. Nulla consequat massa quis
      enim. Donec pede justo, fringilla vel, aliquet nec, vulputate
      eget, arcu. In enim justo, rhoncus ut, imperdiet a venenati</p>
</div>
```

Der Container mit der `id="navigation"` beinhaltet eine Liste mit Navigationspunkten und soll links neben dem eigentlichen Inhalt positioniert werden. Um dies zu erreichen, sollten wir ihm zuerst eine Breite zuweisen, da wir anschließend die Eigenschaft `float` nutzen möchten, um ihn links zu positionieren. Gefloatete Elemente werden aus dem normalen Dokumentenfluss herausgenommen und lassen andere Elemente um sich herum fließen.

In dem Moment, in dem einem Element diese Eigenschaft zugewiesen wird, wird es automatisch zu einem Blockelement, d.h. ihm kann eine Höhe und eine Breite zugewiesen werden. Wenn gefloatete Elemente keine Breite aufweisen, kann das unter Umständen zu unberechenbaren Ergebnissen führen.

```
#navigation
{
border:solid 1px;
width:220px;
float:left;
}
```

Indem der Container mit der `id="inhalt"` ein `margin-left` von 240px aufweist, wird dafür gesorgt, dass der Inhalt nicht unter die Navigationsbox rutscht, wenn diese Box kürzer ist als der eigentliche Inhalt. Neben `margin-left` weist der Container zusätzlich noch die Eigenschaft `margin-top:70px` auf. Diese Eigenschaft wird nun nicht mehr benötigt, da die Oberkante beider Elemente auf gleicher Höhe liegen soll, und wird einfach entfernt.

```
#inhalt
{
border:solid 1px #0000cc;
padding:10px 20px 10px 20px;
margin: 0px 0px 0px 240px
}
```

Möchten Sie später eine Fußzeile unterhalb der beiden Bereiche anordnen, sollten Sie darauf achten, dass Sie das schwebende Verhalten des Navigationscontainers wieder aufheben, ansonsten würden auch nachfolgende Elemente um den gefloateten Container herumfließen.

```
<div id="navigation">
   <ul>
   <li><a href="#">Menulink</a></li>
   <li><a href="#"> Menulink </a></li>
   <li><a href="#"> Menulink </a></li>
   <li><a href="#"> Menulink </a></li>
   <ul>
</div>
<div id="inhalt">
   <h1> Das ist eine Überschrift</h1>
   <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
      Aenean commodo ligula eget dolor. Aenean massa. Cum sociis
      natoque penatibus et magnis dis parturient montes, nascetur
      ridiculus mus. Donec quam felis, ultricies nec,
      pellentesque eu, pretium quis, sem. Nulla consequat massa quis
      enim. Donec pede justo, fringilla vel, aliquet nec, vulputate
      eget, arcu. In enim justo, rhoncus ut, imperdiet a venenati</p>
</div>
<div id="footer">Fußzeile</div>
```

Dies erreichen sie *unter anderem* mit der CSS-Eigenschaft `clear`. `clear` können die Werte `left`, `right` oder `both` zugewiesen werden. Ein Element, das diese Eigenschaft aufweist, ist immer das erste Elemente, das nicht mehr um das gefloatete Element herumfließt und einen Umbruch erzeugt. In unserem Beispiel sähe der Code folgendermaßen aus:

```
#footer {clear:left}
```

`clear` ist nicht die einzige Methode, um dieses Verhalten zu bewirken. Wenn Sie z.B. um die Elemente `navigation` und `inhalt` einen weiteren Container legen, dem sie die Eigenschaft `overflow:hidden` zuweisen, wird die Fußzeile ebenfalls unterhalb der beiden Container angeordnet.

Dies war nur ein sehr einfaches Bespiel. Im Netz finden Sie ein Sammelsurium guter Bespiele auf CSS2-Basis, die Ihnen sicherlich bei komplexen Layouts helfen können.

Die relativ alte, aber immer noch recht übersichtliche Seite `http://www.intensivstation.ch/en/templates/` bietet eine schnelle Übersicht über mögliche Positionierungsmethoden, genau wie `http://little-boxes.de/lb1/IV-css-positionierung.html`.

Auch CSS entwickelt sich immer weiter. Wer sich für die neuen CSS3-Positionierungskonzepte interessiert, findet hier spannende Informationen:

- `http://www.css3.info/introducing-the-flexible-box-layout-module/`
- `http://www.the-haystack.com/2010/01/23/css3-flexbox-part-1/`
- `http://hacks.mozilla.org/2010/04/the-css-3-flexible-box-model/`

3.8 CSS-Hacks und Browserprobleme

CSS-Hacks braucht man immer dann, wenn einzelne Browser den Code nicht so interpretieren, wie sie eigentlich sollten und die Seiten nicht wie gewünscht dargestellt werden.

Sicherlich gibt es in jedem Browser den ein oder anderen Bug, aber das Hauptproblem stellt sicherlich der Internet Explorer bis einschließlich Version 7 dar. Mit Version 8 hat sich einiges getan und Hacks sind hier kaum noch notwendig. Und mit Version 9 wird nun auch endlich CSS3 unterstützt.

Ich werde hier nur einen kurzen Überblick über die wichtigsten Probleme und Lösungen geben. Auf der Seite `http://www.positioniseverything.net` finden Sie weiterführende Informationen rund um Darstellungsfehler und Bugs der gängigen Browser.

3.8.1 Conditional Comments

Um die Internet Explorer in die richtigen Bahnen zu lenken, ist es sinnvoll, ihnen eine eigene CSS-Datei zuzuweisen. Dies hat den großen Vorteil, dass Sie später diese Dateien einfach und unkompliziert entfernen können, wenn sie nicht mehr benötigt werden.

Microsoft ist sich der Fehler sicherlich bewusst und bietet deshalb die so genannten Conditional Comments an. Hierbei handelt es sich um spezielle HTML-Kommentare, die im Seitenkopf eingebunden werden und je nach Browserversion den beinhaltenden Code ausführen.

Bei dieser Methode handelt es sich um eine proprietäre Lösung von Microsoft, alle anderen Hersteller ignorieren diese Anweisung.

Im Beez-Template werden Sie genau diese Anweisungen finden, um eine einheitliche Darstellung auch im Internet Explorer 6 und 7 zu gewährleisten.

```
<!--[if IE 7]>
<link href="<?php echo $this->baseurl ?>/templates/beez5/css/ie7only.
css" rel="stylesheet" type="text/css" />
<![endif]-->
<!--[if lte IE 6]>
<link href="<?php echo $this->baseurl ?>/templates/beez5/css/ieonly.
css" rel="stylesheet" type="text/css" />
<![endif]-->
```

Die erste Anweisung identifiziert den Internet Explorer in der Version 7 und weist ihm seine individuelle CSS-Datei zu.

Die zweite Anweisung identifiziert alle Internet Explorer Versionen, die kleiner oder gleich dem Internet Explorer 6 sind.

Conditional Comments verfügen über so genannte Operatoren, die Sie sicherlich aus der Mathematik kennen. Mit ihnen lassen sich Bedingungen abfragen. In diesem Fall verwende ich lte, *1te* ist die Abkürzung für „less than equal“.

Es gibt aber weitaus mehr. Die folgenden Listen geben Auskunft über die bestehenden Möglichkeiten.

Selektion der Internet Explorer Versionen

!IE	wenn kein Internet Explorer	<!--[if !IE]>
IE	wenn Internet Explorer	<!--[if IE]>
IE 5.5	wenn Internet Explorer Version 5.5	<!--[if IE 5.5]>
IE 6	wenn Internet Explorer Version 6	<!--[if IE 6]>
IE 7	wenn Internet Explorer Version 7	<!--[if IE 7]>
IE 8	wenn Internet Explorer Version 8	<!--[if IE 8]>
IE 9	wenn Internet Explorer Version 9	<!--[if IE 9]>

Operatoren

!	ungleich	<!--[if !(IE 6)]>
lt	lesser than/kleiner als	<!--[if lt IE 6]>
lte	lesser than equal/kleiner gleich	<!--[if lte IE 6]>
gt	greater than/größer als	<!--[if gt IE 6]>
gte	greater than equal/größer gleich	<!--[if gte IE 6]>

3.8.2 Der *-Hack

Der wohl bekannteste CSS-Filter ist sicherlich der *-Hack (Sternchen-Hack). Dieser Filter wird nur vom Internet Explorer 6 und kleiner interpretiert.

HTML ist normalerweise das erste vorkommende Element innerhalb einer Seite. Der Internet Explorer bis zur Version 6 hat jedoch ein nicht sichtbares, weiteres Element, das das HTML-Element umschließt.

Mit dem *-Selektor können Sie so ein HTML-Element ansprechen, das von einem anderen Element umschlossen wird. Da dies nur in den Internet Explorer Versionen 6 und kleiner der Fall ist, zielt dieser Filter genau darauf ab.

Die Anweisung selbst bleibt standardkonformen Browsern verborgen.

```
#all
{width:960px;}
* html #all
{width:900px; }
```

Mit der zweiten Anweisung überschreiben Sie die erste Anweisung, die für alle Browser gilt, lediglich im Internet Explorer 6 und kleiner.

3.8.3 Schon wieder der Internet Explorer – has layout

has layout ist eine Eigenschaft des Internet Explorers, die öfter zu Problemen in der Darstellung führt. Dahinter steckt ein Konzept, wie Elemente ihren Inhalt anzeigen und begrenzen, wie sie sich im Bezug auf andere Elemente verhalten und wie sie auf Benutzerinteraktion reagieren.

Manche Elemente bringen diese Eigenschaft von sich aus mit, andere nicht. Dies kann zu Darstellungsproblemen führen.

Zu den Elementen, die per se *has layout* aufweisen, gehören:

- `html`, `body`
- `table`, `tr`, `th`, `td`
- `img`
- `hr`
- `input`, `button`, `select`, `textarea`, `fieldset`, `legend`
- `iframe`, `embed`, `object`, `applet`

Wenn es während Ihres Gestaltungsprozesses zu Darstellungsfehlern im Internet Explorer kommt, könnte dies an der fehlenden *has layout*-Eigenschaft bestimmter Elemente liegen.

Sie haben die Möglichkeit, has layout zu erzwingen, indem Sie dem betreffenden Element die Eigenschaft `zoom` mitgeben. Die Anwendung dieser Eigenschaft hilft eigentlich immer, wenn die Ursache des Problems im Konzept des Layouts an sich liegt.

Ich verwende diese Methode im Beez-Template ebenfalls, dort werden zum Beispiel verschachtelte Listen in der rechten Spalte nicht korrekt dargestellt, wenn ich nicht

```
#right ul.menu ul
{zoom:1;}
```

setze.

Sie können es jedoch auch hiermit versuchen:

- `position:absolute`
- `height:` *`numerische Angabe`*
- `width:` *`numerische Angabe`*
- `display: inline-block`
- `float: left | right`

Im Internet Explorer 7 erzwingen Sie *has layout* auch mit der Eigenschaft `overflow: hidden|scroll|auto`.

3.9 CSS-Tuning

Immer dann, wenn Sie mit Joomla! eine sehr große Seite mit hohen Zugriffszahlen bauen, kann es wichtig werden, die Ladezeit der Seite so gering wie nur möglich zu gestalten.

Im Beez-Template binde ich mehrere CSS-Dateien ein. Eines für die Positionierung, eines für die Abstände, eines für die Farben und die für die unterschiedlichen Internet Explorer Versionen.

Der Grund dafür liegt in der einfacheren Handhabung für Designer und in der besseren Verständlichkeit. Für jede Datei muss jedoch ein Request an den Server gesendet werden, was die Performance beeinträchtigt.

Manchmal kann es sinnvoll sein, die Inhalte der Dateien (nicht die für den Internet Explorer) in einer großen Datei zusammenzufassen, was zwangsläufig zu einer besonders umfangreichen Datei führt, die an Übersichtlichkeit verliert.

Sonstige Tipps:

- Vermeiden Sie Redundanzen in Ihren CSS-Regeln.
- Entfernen Sie das Semikolon nach jeder letzten Anweisung in einem Anweisungsblock.
- Fassen Sie Wertangaben zusammen:

```
{
margin:0 10px 0 10px;
```

statt:

```
margin-top:0;
margin-right:10px;
margin-bottom:0px;
margin-left:10px
}
```

- Vermeiden Sie zu viele Einrückungen und Leerzeichen.
- Wenn nötig, entfernen Sie Kommentare.
- Sie können Ihre CSS-Datei auch mit Hilfe des freien Tools CSS-Tidy automatisch optimieren: `http://csstidy.sourceforge.net/`.

Copyright