Besondere Anwendungsmöglichkeiten von CSS

Marie-Theres Tschurlovits*
MatrNr: 0125975
Kennzahl: 066 935

Institut für Softwaretechnik und Interaktive Systeme
Gruppe für Interaktive Mediale Systeme (IMS)
Technische Universität Wien

Inhaltsverzeichnis

1 Einleitung

Unter CSS (Cascading Style Sheets, kaskadierende Stylesheets) kann man sich im Grunde Formatvorlagen vorstellen. Vor der Einführung von CSS wurde beispielsweise in HTML (Hypertext Markup Language) die Formatierung innerhalb der HTML-Tags vorgenommen. Jede Änderung des Layouts musste so mühsam an mehreren Stellen durchgeführt werden (z.B. Änderung der Farbe aller `h1`-Überschriften). CSS ermöglicht es an einer globalen Stelle Formatdefinitionen zu platzieren, beispielsweise durch Auslagerung der Layoutdefinition in eine eigene Datei (z.B. `style.css`), was dazu führt, dass sich die Änderung einer Eigenschaft an einer zentralen Stelle wirklich auf das gesamte Dokument, die gesamte Webseite, auswirkt.

Die Formatierung mit Cascading Style Sheets bietet viel größere Gestaltungsfreiräume als das mit der Direktformatierung von HTML-Tags erreicht werden könnte. Außerdem trägt CSS wesentlich zum Erreichen des „Ideals“ des WWW (World Wide Web) bei, den Inhalt und das Layout einer Webseite voneinander zu trennen. So können HTML/XHTML/XML etc. wirklich als die Auszeichnungssprachen (Markup Languages) eingesetzt werden, die sie von der Grundidee her eigentlich sind. [4]

In dieser Arbeit sollen Eigenschaften von CSS erläutert werden, die vielleicht noch nicht so allgemein bekannt sind, wie die einfache Schriftart- oder Farbformatierung. Grundkenntnisse aus HTML und CSS werden dabei vorausgesetzt. Die beigefügten Screenshots wurden mit den Browsern Mozilla Firefox 1.0.4 bzw. Internet Explorer 6.0 für Windows erstellt. Bei der Nennung des Microsoft Internet Explorers ist, solange nichts anderes angegeben, die Windows-Version gemeint.

2 Kaskade

Stylesheets können nicht nur von Autoren, sondern auch von Benutzern und Benutzerprogrammen (User-Agents, z.B. ein Browser) angegeben werden. Diese Stylesheets unterschiedlicher Herkunft weisen einander überlappende Gültigkeitsbereiche auf und arbeiten entsprechend der *Kaskade* (*Cascading* Style Sheets) zusammen. Die Kaskade von CSS gibt jeder Stilregel eine Gewichtung und wählt bei mehreren Regeln diejenige mit der höchsten Priorität aus. [2, S. 605 ff.]

*e0125975@student.tuwien.ac.at

2.1 !important-Regel

Standardmäßig haben Stylesheets von Autoren mehr Gewicht, gefolgt von Benutzerregeln und jenen des Benutzerprogramms. Höchste Priorität erhält eine Regel, die mit `!important` markiert wurde. Bei Einsatz der `!important`-Regel erhält nun das Benutzer-Stylesheet gegenüber dem Stylesheet des Autors eine höhere Priorität. Das ist beispielsweise dann sinnvoll, wenn ein Benutzer mit Sehschwäche den Text einer Seite aufgrund einer zu klein gewählten Schrift des Autors gar nicht lesen könnte. Da sein eventuell vorhandenes Benutzer-Stylesheet vom Stylesheet des Autors überschrieben wird, bietet sich durch diese Regel die Möglichkeit trotzdem eine größere Schrift zu wählen. Solch eine explizite Beeinflussung der Regel-Rangordnung, mit der auch Autoren Werte auf höchste Priorität setzen können, sollte nur mit Maß und Ziel eingesetzt und nicht für gewisse browserspezifische Tricks verwendet werden. [2, S. 605 ff.]

2.2 Kaskaden-Reihenfolge

`body` bzw. `p` sind *Elemente*, `font-size` eine *Eigenschaft* und `blue`, `yellow` bzw. `red` die konkreten *Werte*.

Beispiel CSS

```
body {color: blue;}
p {color: yellow;}
p {color: red;}
```

Beispiel HTML

```
<html>
<head>
   <link rel="stylesheet"
   type="text/css" href="style.css">
</head>
<body>
   Testtext im body!
   <p>
      Testtext im 1. Absatz!
   </p>
</body>
</html>
```

Um nun den tatsächlichen Wert einer Element/Eigenschafts-Kombination zu ermitteln wenden die Benutzerprogramme eine vorgegebene Reihenfolge an.

1. Es werden alle Regeln ermittelt, die sich auf das entsprechende Element und dessen Eigenschaft beziehen. Eine Regel wird angewandt, wenn der entsprechende *Selektor* (entspricht dem Element vor den geschweiften Klammern einer CSS-Regel, in diesem Fall `p`) mit dem Element (hier: `<p>`) übereinstimmt.

2. Es erfolgt eine erste Sortierung nach der oben angesprochenen Gewichtung: Autoren-Stylesheets vor Benutzer-Stylesheets vor Stylesheets der Benutzerprogramme vor Standard-Stylesheets. Im Falle von `!important`-Regeln überschreiben die Stylesheets der Benutzer diejenigen der Autoren.

3. Bei der nächsten Sortierung überschreiben spezifischere Selektoren (z.B. `p`) allgemeinere Selektoren (z.B. `body`).

4. Letztendlich kann sich noch die angegebene Reihenfolge auf das Endergebnis auswirken. Wenn zwei Regeln gleiche Gewichtung, Ursprung und Spezifität haben, hat die zuletzt angegebene Regel höhere Priorität. Im obigen Beispiel überschreibt also der zuletzt angegebene Wert für die Farbe des `p`-Elements, `red`, die vorherigen Farbangaben. (Abbildung 1, 2)

3 Kurzformen von CSS-Regeln (CSS shorthand property)

CSS bietet einerseits die Möglichkeit Teileigenschaften (z.B. oberer Rand = `margin-top` und rechter Rand = `margin-right`) getrennt zu formatieren oder auch gemeinsam in einem einzigen Befehl (*CSS shorthand property*).

Je nach Anwendung kann die eine oder die andere Variante oder auch eine Kombination davon sinnvoll sein. Werden beispielsweise immer alle Teileigenschaften angeben, kann ein einzelner Befehl, der alle Kommandos zusammenfasst, übersichtlicher sein. Für den ungeübten Anwender könnten zu Beginn größere Verständnisschwierigkeiten entstehen, da nur mehr die allgemeine Eigenschaft gefolgt von einem Doppelpunkt und den durch Kommas getrennten Werten notiert wird und die Namen der Teileigenschaften ausgespart werden.

Außerdem ist zu beachten, dass die momentan aktuellen Versionen der Browser diese Schreibweise bereits unterstützen, man bei ältern Browsern oft auf fehlende bzw. nur teilweise Unterstützung dieser Eigenschaft trifft. [1]

Diese Form der Kurzschreibweise ist für mehrere CSS-Eigenschaften verfügbar:

- `font`: font-weight, font-style, font-variant, font-size, line-height, font-family

Abbildung 1: Kaskaden-Reihenfolge – Firefox 1.0.4 (FF)

Abbildung 2: Kaskaden-Reihenfolge – Internet Explorer 6.0 (IE)

- `background`: background-color, background-image, background-repeat, background-attachment, background-position
- `margin`: margin-top, margin-right, margin-bottom, margin-left
- `padding`: padding-top, padding-right, padding-bottom, padding-left
- `border`: border-width, border-style, border-color
- `border-width`: border-top-width, border-right-width, border-bottom-width, border-left-width
- `border-style`: border-top-style, border-right-style, border-bottom-style, border-left-style
- `border-color`: border-top-color, border-right-color, border-bottom-color, border-left-color
- `list-style`: list-style-type, list-style-position, list-style-image
- `outline`: outline-color, outline-style, outline-width

Im Folgenden wird die Kurzschreibweise für CSS-Eigenschaften anhand einiger Beispiele demonstriert.

3.1 font

Formatierung mit mehreren Einzel-Befehlen:

```
font-size: 1em;
line-height: 1.5em;
font-weight: bold;
font-style: italic;
font-variant: small-caps;
font-family: verdana, sans-serif;
```

Formatierung mit einem Kurzbefehl:

```
font: 1em/1.5em bold italic small-caps
   verdana, sans-serif;
```

Diese Zusammenfassung kann, wenn immer sehr viele Eigenschaften der Fonts angegeben werden zu einer Erhöhung der Übersichtlichkeit des CSS-Codes führen. Damit die Interpretation dieser Kurzschreibweise in jedem Fall korrekt erfolgt, sind noch einige Punkte zu berücksichtigen:

- Sowohl `font-size` als auch `font-family` müssen angegeben werden.
- Werden keine Werte für `font-weight`, `font-style` und `font-variant` festgelegt, werden diese Werte automatisch auf den Standardwert `normal` gesetzt. [2, S. 741 ff.], [3]

3.2 margin, border, padding

Ebenso wie bei der Spezifizierung der Fonts kann auch die Angabe der Eigenschaften von Rändern (margin), Rahmen (border) und Polsterung (padding) in Kurzschreibweise erfolgen. Dies soll im Folgenden an der Eigenschaft der Rahmenbreite (`border-width`) illustriert werden.

Mit den Eigenschaften `border-top-width`, `border-right-width`, `border-bottom-width` und `border-left-width` können die jeweiligen Rahmenbreiten einzeln angegeben werden. Mit der Eigenschaft `border-width` können die Angaben aller Rahmen zusammengefasst werden.

Wird nur ein Wert angeben, gilt dieser für alle Seiten, d.h. für alle vier Rahmen. Gibt man zwei Werte an, getrennt durch ein Leerzeichen, so gilt der erste für den oberen und den unteren, der zweite für den rechten und den linken Rahmen. Bei der Angabe von drei Werten bestimmt der erste Werte den oberen, der zweite den rechten und den linken und der dritte Wert den unteren Rahmen. Gibt man vier Werte an werden die Rahmen in folgender Reihenfolge gesetzt: oben, rechts, unten, links. [2, S. 613 ff.]

Beispiel CSS

```
p {color: red; border: solid;}

/* Alle Rahmen: thin */
p#eins {border-width: thin;}

/* top/bottom: thin, right/left:
   thick */
p#zwei {border-width: thin thick;}

/* top: thin, right/left: thick,
   bottom: medium */
p#drei {border-width: thin thick
                      medium;}

/* top: thin, right: thick,
   bottom: medium, left: thin */
p#vier {border-width: thin thick
                      medium thin;}
```

Beispiel HTML

```
<p id="eins">blah</p>
<p id="zwei">blah</p>
<p id="drei">blah</p>
<p id="vier">blah</p>
```

Abbildung 3: Rahmen (border) (FF)

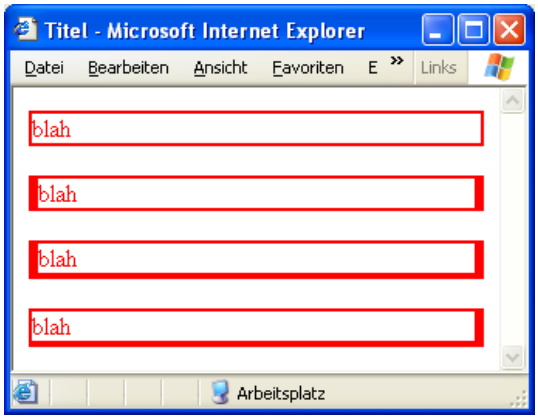

Abbildung 4: Rahmen (border) (IE)

An diesem Beispiel (Abbildung 3, 4) ist außerdem zu sehen, dass die verschiedenen Browser manche Eigenschaften etwas unterschiedlich interpretieren. In diesem Fall unterscheiden sich die Standardwerte der Rahmenbreiten des Firefox-Browsers und des Internet Explorers für `thin`, `thick` und `medium`.

Diese Standardwerte werden deshalb gewählt, da bei der Angabe des Rahmens nur der Rahmentyp angegeben wurde: `border: solid`. Eine Kurzschreibweise für die Angabe des Rahmens

```
border: border-width border-style
        color;
```

wäre beispielsweise

```
border: 1px solid black;
```

was einen einfachen schwarzen Rahmen mit einem Pixel Breite erzeugen würde. Gibt man nur den Stil des Rahmens an (hier: `solid`) werden für Rahmenbreite und -farbe Standardwerte genommen. Der Standardwert für die Rahmenbreite ist `medium` und die Farbe des Rahmens wird von der Farbe des eingeschlossenen Textes bestimmt (hier: `red`). [2, S. 627], [3]

4 Fixe Positionierung

4.1 Problematik

Ein oft gehegter Wunsch ist es, einen bestimmten Teil einer Webseite, meist ein Logo oder ein Menü, an einer fixen Position im Browserfenster erscheinen zu lassen. Auch wenn die Seite gescrollt wird, soll dieser Bereich fix an seiner Position, z.B. der linken oberen Ecke des Browserfensters, bleiben. In der Regel wird dieser Effekt durch die Verwendung von Frames erreicht. Leider bringt die Verwendung von Frames auch einige Nachteile mit sich.

Einige ältere Browser unterstützen Frames überhaupt nicht. Der Autor der Webseite kann zwar mit Hilfe des `<noframes>`-Tags Informationen für Browser bereitstellen, die keine Frames unterstützen. In der Praxis wird diese Möglichkeit aber kaum bzw. unzureichend genutzt. Meist enthält dieser Tag nur einen kurzen Absatz, dass der Browser keine Frames unterstützt, oder einen Link, der zum Download eines framefähigen Browsers führt. Auch wenn es für Personen, die regelmäßig mit dem Web arbeiten oft selbstverständlich ist, einen halbwegs modernen Browser zu verwenden, kann dies nicht vorausgesetzt werden und führt zu einem Ausschluss eines Teils der Web-User.

Bei älteren Browsern gab es große Probleme hinsichtlich des korrekten Setzens von Bookmarks (Lesezeichen) von Unterseiten, die in ein Frameset geladen wurden. Zwar wurden diese Probleme teilweise gelöst oder zumindest verbessert, jedoch haben Frames nach wie vor den Nachteil, dass sie zwar einerseits eine optische Gliederung (des Layouts) erzeugen, andererseits aber auch den eigentlichen Inhalt aufsplitten. Beispielsweise ist ein Verlinken auf eine Unterseite eines Framesets, sodass diese Seite nicht für sich allein stehend sondern innerhalb des ursprünglichen Framesets erscheint, nicht ohne einen gewissen Mehraufwand in Form kleiner „Tricks“ möglich.

Aufgrund der oben genannten Probleme ist es heutzutage in vielen Fällen nicht mehr angebracht Frames zu verwenden. Die fixe Positionierung eines Teilbereichs der Seite hätte eigentlich viel leichter mit dem für HTML 3.0 vorgeschlagenen `<banner>`-Tag erreicht werden können, der jedoch nicht in die Spezifikation aufgenommen wurde. [5]

CSS bietet nun eine Möglichkeit, einen Teilbereich einer Seite (z.B. ein Menü) fix zu positionieren, die nicht die oben genannten Nachteile von Frames mit sich bringt. Diese Nachteile entstehen erst gar nicht, da CSS ja ledig-

lich das Layout der Seite verändert, nicht den eigentlichen Inhalt. Wären Stylesheets im Browser deaktiviert, wäre die Seite zwar optisch sicher nicht mehr so reizvoll, der Funktionsumfang der Seite würde dadurch aber nicht beeinträchtigt.

4.2 Positionierungsschemata

Um das oben genannte Problem zu lösen greift man auf eine Form der absoluten Positionierung einer Box zurück. Bei der absoluten Positionierung wird eine Box (z.B. `<div>`, `<p>`, `<h1>`, ...) vollständig aus dem normalen Fluss entfernt. Das bedeutet, dass diese Box keinen Einfluss auf spätere gleichrangige Elemente hat. Der Box wird nun eine Position relativ zu der sie umschließenden Box zugewiesen. Die CSS-Eigenschaft `position` kann folgende Werte annehmen:

- `static`: Die Box befindet sich im normalen Fluss. Die Eigenschaft `left` und `top` sind nicht verfügbar.
- `relative`: Die ursprüngliche Position der Box wird entsprechend dem normalen Fluss berechnet. Danach wird die Box entsprechend der Wertangaben relativ zu ihrer ursprünglichen Position weiter verschoben. Für darauf folgende Elemente verhält es sich so, als wäre diese Box nie verschoben worden.
- `absolute`: Die Position und eventuell die Größe der Box wird mit den Eigenschaften `left`, `right`, `top`, `bottom` (bzw. `width`, `height`) oder einer Teilmenge aus diesen angegeben. Dabei werden die Abstände relativ zu dem umschließenden Block bestimmt. Mit dem Wert `absolute` positionierte Boxen werden aus dem normalen Fluss entfernt und haben deshalb keinen Einfluss auf das Layout späterer gleichrangiger Elemente.
- `fixed`: Die Position wird wie im absolute-Modell berechnet, erhält aber darüber hinaus die Eigenschaft, hinsichtlich eines Bezugspunktes fixiert zu sein.
  - *Endlosmedien*: z.B. eine HTML-Seite: Hier ist die mit fixed positionierte Box feststehend zum Viewport (dem Browserfenster) und bewegt sich beim Scrollen der Seite nicht mit.
  - *Seitenmedien*: z.B. Druckvorschau: Die Box ist relativ zu den einzelnen Seiten fixiert und erscheint beispielsweise am Beginn jeder neuen Seite.

Mit Hilfe der `@media`-Regel kann festgelegt werden, um welches Ausgabemedium es sich handelt. So kann beispielsweise erreicht werden, dass ein Logo sich zwar auf der HTML-Seite immer fix am oberen Browserfensterrand befindet, in einem Ausdruck aber nur auf der ersten Seite erscheint. [2, S. 636 ff.]

4.3 Browserkompatibilität

Auf den ersten Blick scheint es, dass das Problem der fixen Positionierung mit Hilfe von CSS nun endgültig gelöst werden könnte. Leider scheitert das daran, dass noch nicht alle gängigen Browser diese CSS-Eigenschaft unterstützen. Während aktuelle Versionen der Browser Mozilla/Firefox, Netscape und Opera diese Eigenschaft unterstützen, versteht die aktuelle Version des Internet Explorer (6.0) diese Eigenschaft noch nicht. Die Mac-Version (Internet Explorer 5.2) kann jedoch bereits mit `position: fixed` umgehen. Eine Möglichkeit das Problem der Kompatibilität zu umgehen ist es, den Wert `fixed` solange nicht zu verwenden, bis er auch vom Internet Explorer, dem meistgenutzten Browser, verstanden wird.

Eine andere Möglichkeit ist es, durch eine Form der „Browser-Weiche“ verschiedene Stylesheets für die diversen Browser anzugeben. Man kann dies auf verschiedene Arten erreichen, z.B. mit einer JavaScript-Abfrage, mit „Conditional Comments“ oder unter Ausnutzung von Eigenschaften, die bestimmte Browser gar nicht verstehen um so gewisse CSS-Formatierungen vor ihnen zu verstecken. [8]

4.4 Lösungsansatz Conditional Comments

Um ein eigenes Stylesheet für den Internet Explorer einzubinden scheinen die „Conditional Comments“ eine ansprechende Lösung zu sein. Zum ersten ist dafür kein im Browser aktiviertes JavaScript von Nöten, zum zweiten verlässt man sich bei diesem Lösungsweg auch nicht auf mangelhafte CSS-Unterstützung oder Bugs bestimmter Browser, was oft zu fehlerhaften Ergebnissen mit unerwünschten Effekten führen kann. Microsoft erlaubt es, ab der Version 5 des Internet Explorers, Bedingungen innerhalb von Kommentar-Tags anzugeben.

```
<link rel="stylesheet" type="text/css"
   href="style/style.css">
<!--[if gte IE 5]>
   <link rel="stylesheet"
      type="text/css"
      href="style/style_ie.css">
<![endif]-->
```

Zuerst wird der Verweis auf das externe Style-Sheet `style.css` gelesen. Darauf folgt, innerhalb von HTML-Kommentaren `<!-- -->`, eine Abfrage, ob es sich bei dem Browser um einen Internet Explorer der Version größer als 5 handelt. Nur Browser, die diese Eigenschaften erfüllen, lesen die nächste Zeile mit dem Verweis auf das Stylesheet `style_ie.css`, in dem die Internet Explorer spezifischen CSS-Eigenschaften codiert sind. Diese Abfrage innerhalb eines HTML-Kommentars wird nur vom Internet Explorer gelesen und von allen anderen Browsern ignoriert.

So ist es auf einfache Weise möglich ein Stylesheet mit der Eigenschaft `position: fixed` für die Browser anzugeben, die sie verstehen und für den Internet Explorer ein eigenes Stylesheet bereitzustellen, wo stattdessen `position: absolute` codiert ist.

5 Medientypen

Stylesheets bieten die Möglichkeit zu spezifizieren wie ein Dokument auf verschiedenen Medientypen (Bildschirm, Papier, Sprachsynthesizer, Braille-Gerät, ...) ausgegeben werden soll.

Abbildung 5: Darstellung am Monitor (FF)

Abbildung 6: Darstellung in der Druckvorschau (FF)

Einerseits gibt es Eigenschaften, die nur für bestimmte Medien vorgesehen sind, andererseits können auch für mehrere Medien dieselben Eigenschaften, jedoch mit unterschiedlichen Werten, eingesetzt werden. Ein Beispiel für letzteres wäre die Angabe von unterschiedlichen Werten für die Schriftgröße (`font-size`) und die Schriftart (`font-family`) für ein Printmedium und die Anzeige auf einem Monitor (Abbildung 5-8).

Es gibt zwei Möglichkeiten für die Angabe medienabhängiger Stylesheets die im Folgenden beschrieben werden. [2, S. 610 ff.]

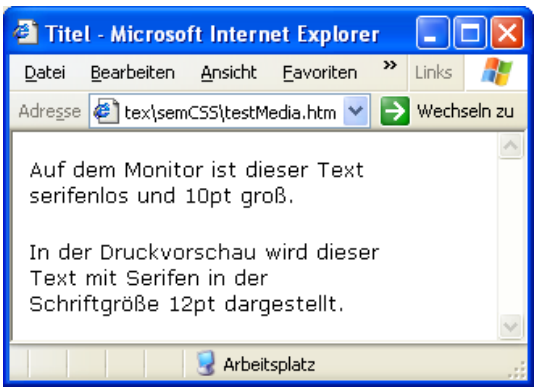

Abbildung 7: Darstellung am Monitor (IE)

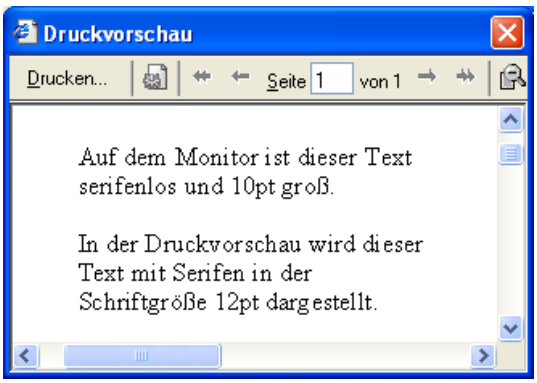

Abbildung 8: Darstellung in der Druckvorschau (IE)

5.1 @media, @import

Medienabhängige Stylesheets können mit Hilfe der Regeln `@media` oder `@import` angegeben werden:

```
/* Stylesheet für Sprachausgabe */
@import url("sprache.css") aural;

@media print
{
    /* Stylesheet für Druck-
       ausgabe */
}
```

Bei der `@media`-Regel können Stylesheet-Regeln für verschiedene Medien in einem einzigen Stylesheet angegeben werden. Auch die Angabe mehrerer Zielmedien, durch Kommas getrennt, ist möglich. [2, S. 610 ff.]

```
/* Schriftart und -größe für
   Druck */
@media print
{
    body
    {
        font-family: serif;
        font-size: 12pt;
```

```
    }
}

/* Schriftart für Monitor */
@media screen
{
    body
    {
        font-family: sans-serif;
        font-size: 10pt;
    }
}

/* Zeilenhöhe für Druck und
   Monitor */
@media screen, print
{
    line-height: 1em;
}
```

Die `@import`-Regel ermöglicht es, Formatierungsregeln aus anderen Stylesheets zu importieren. Dabei ist zu beachten, dass die `@import`-Regeln vor allen anderen Regelmengen im Stylesheet stehen müssen. Dem Schlüsselwort `@import` folgt die URI (URL) des Stylesheets, das eingebunden werden soll. Es gibt zwei äquivalente Syntaxformen:

```
@import "style.css";
@import url("style.css");
```

Es besteht weiters die Möglichkeit, die `@import`-Regeln medienabhängig zu gestalten. Dabei werden hinter der URI des Stylesheets die durch Kommas getrennten Medientypen aufgelistet. Diese `@import`-Regel hat dieselbe Wirkung als wäre das Stylesheet in eine `@media`-Regel für dasselbe Medium eingeschlossen worden. Der Unterschied zur `@media`-Regel besteht darin, dass das Benutzerprogramm nur diejenigen Stylesheets herunterlädt, die für das jeweilige Medium wirklich benötigt werden. Wird kein Medientyp bzw. der Medientyp `all` angeben, werden alle Stylesheets importiert. [2, S. 604 ff.]

5.2 media-Attribut

In HTML 4.0 kann beispielsweise über das `media`-Attribut des `link`-Tags das Zielmedium eines externen Stylesheets angeben werden:

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

/* Stylesheet für Druckausgabe */
<link rel="stylesheet"
      type="text/css"
      media="print"
      href="print.css">
```

Es können auch mehrere Ausgabemedien dasselbe Stylesheet zugewiesen bekommen: [2, S. 610 ff.]

```
/* Stylesheet für Bildschirm- und
   Druckausgabe */
<link rel="stylesheet"
      type="text/css"
      media="screen, print"
      href="style.css">
```

6 Zentrieren eines Blockelements

Lange Zeit war es in HTML gebräuchlich ein Blockelement auf folgende Weise auszurichten:

```
/* Blockelement zentrieren */
<div align="center">
    ...
</div>
```

Abbildung 9: Zentriertes Blockelement Variante 1 (FF)

Abbildung 10: Zentriertes Blockelement Variante 1 (IE 6.0 Quirks-Modus)

Jedoch sind HTML-Präsentationsattribute seit der Einführung von Stylesheet-Alternativen *missbilligt* (*deprecated*). Das bedeutet, dass vorzugsweise Stylesheets für Stil- und Gestaltungseffekte eingesetzt werden sollten. Die Benutzerprogramme sollten diese Attribute aufgrund der Abwärtskompatibilität weiter unterstützen.

In CSS lässt sich das Zentrieren eines Blockelements jedoch nicht so direkt erreichen wie mit HTML. Außerdem zeigt sich hier unterschiedliches Verhalten der Browser. Bis zur Version 6.0 des Internet Explorers wurde ein

Abbildung 11: Zentriertes Blockelement Variante 1 (IE 6.0)

Blockelement, unter Angabe der folgenden Eigenschaften, nicht zentriert.

Beispiel HTML

```
<p id="content">
    Dieser Block sollte zentriert
    erscheinen!
</p>
```

Beispiel CSS Variante 1

```
#content
{
    width: 150px;
    margin-left: auto;
    margin-right: auto;
    border: 1px solid black;
}
```

Für den Internet Explorer (Versionen vor 6.0) war ein kleiner Trick notwendig, der auch von den anderen Browsern korrekt interpretiert wird. Im `body` musste eine zentrierte Textausrichtung angegeben werden, die im Blockelement mit einer linksbündigen Ausrichtung des Textes überschrieben wurde. Das Ergebnis war, wie gewünscht, ein zentriertes Blockelement mit linksbündig ausgerichtetem Text. (Abbildung 9-13)

Dieser Umstand ist auch deshalb von Bedeutung, da auch ein Internet Explorer 6.0 im so genannten „Quirks-Mode“ diese Umgehungsstrategie zur korrekten Anzeige benötigt. (Abbildung 10) [1], [3], [9], [10]

Abbildung 12: Zentriertes Blockelement Variante 2 (FF)

Abbildung 13: Zentriertes Blockelement Variante 2 (IE)

Beispiel CSS Variante 2

```
/* Zentriertes Blockelement für Pre-
   Internet Explorer 6.0 Versionen */
body
{
    text-align: center;
}

#content
{
    text-align: left;
    width: 150px;
    margin-left: auto;
    margin-right: auto;
    border: 1px solid black;
}
```

7 Hintergrundfarbe bis zum Rand des Browserfensters

Baut man das Grundgerüst für das Layout einer Webseite mit Hilfe von Tabellen oder Frames auf, ist es ein leichtes beispielsweise eine Spalte, die das Menü enthält, in einer anderen Hintergrundfarbe als die restliche Seite, bis zum unteren Rand des Browserfensters laufen zu lassen.

Abbildung 14: Hintergrundfarbe endet nach Blockende (FF)

Ein Nachteil von CSS ist momentan, dass es in vertikaler Richtung nicht gut kontrollierbar ist. Das bedeutet in

Abbildung 15: Hintergrundfarbe endet nach Blockende (IE)

diesem Fall, dass man zwar eine Box mit entsprechender Hintergrundfarbe, Ausrichtung und Größe erzeugen kann, die Box aber in vertikaler Richtung nur so groß ist, wie der sich darin befindliche Inhalt. (Abbildung 14, 15)

Beispiel HTML

```
<div id="nav">
    <ul>
        <li>Punkt 1</li>
        <li>Punkt 2</li>
        <li>Punkt 3</li>
    </ul>
</div>
```

Beispiel CSS

```
#nav
{
    background-color: #CCCCCC;
    width: 110px;
}
```

Abbildung 16: Hintergrundfarbe reicht bis Rand des Browserfensters (FF)

Natürlich kann man durch Angabe der Höhe der Box eine bestimmte vertikale Ausdehnung erreichen, nur gibt es auf den ersten Blick keine Möglichkeit, die Box einfach bis zum unteren Rand des Browserfensters laufen zu

Abbildung 17: Hintergrundfarbe reicht bis Rand des Browserfensters (IE)

lassen, sodass die Box auch beim Scrollen, Änderung der Fenstergröße und verschiedenen Auflösungen, immer erst dort endet. Durch einen kleinen Trick lässt sich dieses Problem jedoch lösen. Dazu fügt man ein Hintergrundbild (`background-image`) in exakt der gewünschten Farbe und Breite wie der gewünschten Box ein. (Abbildung 16, 17)

```
body
{
    background-image: url("grey.gif");
    background-repeat: repeat-y;
}
```

Außerdem legt man fest, dass sich das Hintergrundbild über die gesamte Höhe des Elements (`body`) wiederholt (`background-repeat: repeat-y`).

Zwar wird die Eigenschaft `background-repeat: repeat-y` von vielen Browsern unterstützt, der Nachteil dieser Methode ist jedoch, dass die Breitenangabe der Menübox nicht mit relativen Größenangaben wie `em` gemacht werden sollte, da der Benutzer zwar die Textgröße und damit die Breite der Box verändern könnte, die Breite der Hintergrundfarbe sich aber aufgrund der Tatsache, dass sie durch ein Bild mit fixer Breite simuliert wird, nicht verändern würde. [3], [7]

8 Erzeugter Inhalt

Manchmal kann es sinnvoll sein, gewisse Inhalte automatisch erzeugen zu lassen. Gute Beispiele dafür wären die automatische Nummerierung von Listen oder Kapitelüberschriften oder das Einfügen des Wortes „Abbildung“ vor jeder Bildunterschrift.

In CSS kann Inhalt auf folgende Arten generiert werden:

- Durch die Eigenschaft `content`, kombiniert mit den Pseudoelementen `:before` und `:after`.
- Mit Hilfe der akustischen Eigenschaften `cue-before` und `cue-after`.

- Mit dem Wert `list-item` für die Eigenschaft `display`.

8.1 Pseudoelemente :before und :after

Durch die Pseudoelemente `:before` und `:after` kann Inhalt automatisch *vor* oder *nach* dem Inhalt eines Elements eingefügt werden. Mit der Eigenschaft `content` wird der einzufügende Inhalt angegeben.

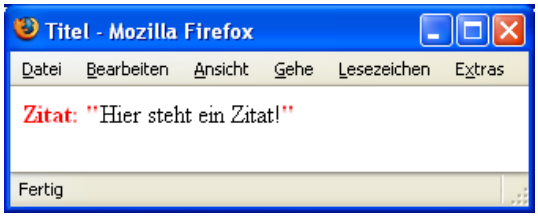

Abbildung 18: Erzeugter Inhalt mit den Pseudoelementen :before und :after (FF)

Abbildung 19: Erzeugter Inhalt mit den Pseudoelementen :before und :after (IE)

In folgendem Beispiel werden mit Hilfe von CSS vor und nach dem Inhalt eines p-Elements automatisch Anführungszeichen eingefügt:

Beispiel CSS

```
p.cite:before
{
    color: red;
    font-weight: bold;
    content: "Zitat: \"";
}

p.cite:after
{
    color: red;
    font-weight: bold;
    content: "\"";
}
```

Beispiel HTML

```
<p class="cite">Hier steht ein
    Zitat!</p>
```

Diese praktischen Pseudoelemente werden momentan von allen aktuellen Browsern, bis auf den Internet Explorer für Windows (siehe Abbildung 18, 19), unterstützt. [2, S. 684 ff.], [6]

8.2 Automatische Zähler und Nummerierung

In CSS können mit den Eigenschaften `counter-increment` und `counter-reset` automatisch Zähler und Nummerierungen erzeugt werden. Die durch diese beiden Eigenschaften definierten Zähler werden in Kombination mit den Funktionen `counter()` bzw. `counters()` der Eigenschaft `content` verwendet.

Abbildung 20: Automatische Nummerierung von Kapitelüberschriften (FF)

Abbildung 21: Automatische Nummerierung von Kapitelüberschriften (IE)

Die Eigenschaft `counter-increment` akzeptiert einen oder mehrere Namen von Zählern, optional gefolgt von einer ganzen Zahl, die angibt um wie viel der Zähler bei einem Inkrementierungsschritt erhöht wird.

Der Eigenschaft `counter-reset` wird eine Liste mit einem oder mehreren Namen von Zählern übergeben, ebenfalls optional gefolgt von einer ganzen Zahl, die den Wert bestimmt, auf den der Zähler bei einem Reset gesetzt wird.

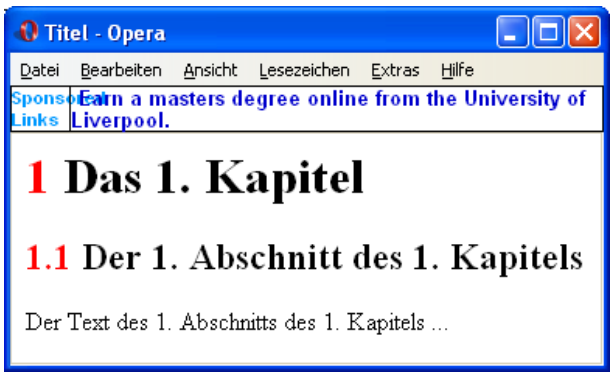

Abbildung 22: Automatische Nummerierung von Kapitelüberschriften (Opera 8.0)

Das folgende Beispiel zeigt, wie Kapitel und Abschnitte innerhalb dieser Kapitel automatisch nach dem Schema „Kapitel 1“, „Kapitel 1.1“, „Kapitel 1.2“, ... durchnummeriert werden können:

Beispiel CSS

```
h1:before
{
   /* Kapitelnr */
   content: counter(chapter) " ";

   /* addiere 1 zu 'chapter' */
   counter-increment: chapter;

   /* setze 'section' auf 0 */
   counter-reset: section;
   color: red;
}

h2:before
{
   /* Kapitelnr.Abschnittnr */
   content: counter(chapter) "."
            counter(section) " ";

   /* addiere 1 zu 'section' */
   counter-increment: section;
   color: red;
}
```

Beispiel HTML

```
<h1>Das 1. Kapitel</h1>
<h2>Der 1. Abschnitt des
   1. Kapitels</h2>
<p>Der Text des 1. Abschnitts des
   1. Kapitels ...</p>
```

Jedes Mal, wenn in obigem Beispiel eine `h1`-Überschrift angezeigt werden soll, wird der eigentlichen Kapitelüberschrift die Nummer des aktuellen Kapitels vorangestellt. Danach wird der Zähler `chapter` für das Kapitel erhöht. Ähnlich verhält es sich mit einer `h2`-Überschrift.

Wie auch aus den Abbildungen 20-22 ersichtlich, werden die nützlichen Eigenschaften `counter-increment` und `counter-reset` von den aktuellen Browsern, Ausnahme Opera, noch nicht unterstützt. [2, S. 693 ff.], [6]

9 Literaturtipps

- Die offizielle und wahrscheinlich aktuellste Informationsquelle zu CSS ist die *Cascading Style Sheets Home Page* des W3C (World Wide Web Consortium): `http://www.w3.org/Style/CSS`
- *XHTML, CSS & Co* [2] ist die deutsche Übersetzung der W3C-Spezifikation zu HTML 4.01, XHTML 1.0, XHTML 1.1 und Cascading Style Sheets, Level 2, in gedruckter Form (auch als e-Book erhältlich)
- *CSS 4 You* ist laut seiner Autoren eine „nahezu komplette CSS-Referenz auf deutsch“ und bietet auch eine gute Übersicht über die Browser-Kompatibilität von CSS [6]: `http://www.css4you.de`
- *SELFHTML* bietet eine gute und übersichtliche Einführung und Tipps & Tricks für den Einsatz von CSS: `http://de.selfhtml.org/css`

10 Zusammenfassung

In dieser Arbeit wurde eine kleine Auswahl interessanter Möglichkeiten, die CSS bietet, vorgestellt, die wahrscheinlich noch keinen so großen Bekanntheitsgrad erreicht haben dürften. Es wurde auch versucht aufzuzeigen, wie vielfältig und nützlich die Möglichkeiten der Gestaltung von Webseiten mit CSS sind. Mit besserer und einheitlicherer Browserunterstützung und Weiterentwicklung der Cascading Style Sheets (Level 3 ist in Arbeit) trägt der zunehmende Einsatz von CSS hoffentlich dazu bei, die Zahl der qualitativ hochwertigen, einheitlich gestalteten und leichter zugänglichen Webseiten (Stichwort accessibility) in den Weiten des Internets zu erhöhen.

Diese Arbeit erhebt keinerlei Anspruch auf Vollständigkeit. CSS bietet noch viele weitere interessante Möglichkeiten, die nicht alle aufgenommen werden konnten und die den Umfang dieser Arbeit bei weitem sprengen würden.

Literatur

[1] Tantek Celik. *Ten CSS tricks – corrected and improved*. http://tantek.com/log/2004/09.html#d07t1434, 10.09.2004 (Stand: 02.06.2005).

[2] Stefan Mintert. *XHTML, CSS & Co: Die W3C-Spezifikation für das Web-Publishing*. Addison-Wesley, München, 2003.

[3] Trenton Moss. *Ten CSS tricks you may not know*. http://www.evolt.org/article/Ten_CSS_tricks_you_may_not_know/17/60369/, 01.09.2004 (Stand: 20.05.2005).

[4] Jakob Nielsen. *Effective Use of Style Sheets*. http://www.useit.com/alertbox/9707a.html, 01.07.1997 (Stand: 02.06.2005).

[5] Jakob Nielsen. *Style Sheets vs. Frames as Web Extensions*. http://www.useit.com/alertbox/styles_vs_frames.html, 07.1997 (Stand: 20.05.2005).

[6] Thomas Pehlgrim. *CSS 4 You - The Finest in Stylesheets: Browser-Kompatibilität*. http://www.css4you.de/browsercomp.html, 28.03.2005 (Stand: 02.06.2005).

[7] Thomas Pehlgrim. *background-repeat: Hintergrund wiederholen: CSS-Referenz auf CSS 4 You - The Finest in Stylesheets*. http://css4you.de/background-repeat.html, 28.03.2005 (Stand: 03.06.2005).

[8] Thomas Pehlgrim. *CSS 4 You - The Finest in Stylesheets: Workshop: Browserweiche*. http://www.css4you.de/wsbw/#bwiecc, 28.03.2005 (Stand: 21.05.2005).

[9] Redaktion SELFHTML. *SELFHTML: Stylesheets / CSS-Formate definieren / Das CSS Box-Modell*. http://de.selfhtml.org/css/formate/box_modell.htm#doctype_switch, 27.10.2001 (Stand: 02.06.2005).

[10] W3C. *Alignment, font styles, and horizontal rules in HTML documents*. http://www.w3.org/TR/REC-html40/present/graphics.html#h-15.1.2, 24.12.1999 (Stand: 02.06.2005).