Cascading Style Sheets

Dokument erstellt am: 1999
Letztmalig geändert am: 31. Mai 2006

1. Einführung: Sinn und Zweck

Mit den Tags der HTML-Spezifikation 4.0 lassen sich zwar Auszeichnungen für bestimmte Textabschnitte angeben, aber von wenigen Ausnahmen abgesehen sind keine Informationen über das Aussehen enthalten. Es ist auch gar nicht gewünscht, dass dies so ist: Wenn man, wie in der Vergangenheit üblich, jede individuelle Auszeichnung über das `<font>`-Tag bewerkstelligen würde, hätte man letzten Endes einen sehr unübersichtlichen HTML-Code – bei ohnehin geringem Umfang an Gestaltungsmöglichkeiten. Weitaus schlimmer würde sich der Versuch gestalten, die getroffenen Einstellungen zu ändern: dies bedeutet dann einen immensen Aufwand.

Deshalb bestreiten alle Drucksatz- und Textverarbeitungsprogramme einen anderen Weg: Inhalt und Aussehen werden getrennt, das Aussehen wird in so genannten Format- oder Stilvorlagen definiert.

Gemeinsame zentrale Stilvorlagen unterstützen zudem die Realisierung eines Corporate Designs, d.h. ein identisches Aussehen aller Seiten einer Site, das sich zudem im Handumdrehen verändern lässt.

Die Möglichkeiten von Vorlagenangaben gehen aber weit über die Definitionsmöglichkeiten der bekannten Attribute der HTML-Tags oder der rein optische Ausgabe eines Dokuments hinaus: Elemente können frei positioniert werden, Text lässt sich mehrspaltig formatieren oder es kann auch die akustische Wiedergabe mit Stilvorlagenvereinbarungen eingestellt werden.

Seit den 4-er Versionen der Browser Netscape und Internet Explorer beherrschen diese Cascading Style Sheets (CSS), wenn auch in unterschiedlichem Umfang. Man kommt so nicht ganz um die Kontrolle des Aussehens umhin.

Eine weitere Idee der Cascading Style Sheets liegt im Kaskadieren: eine neuere Stilanweisung überschreibt eine ältere und – was logisch dasselbe ist – eine überhaupt vorgenommene Stilanweisung überschreibt die Standardvorgabe der Browser. Bei mehrfach vorgenommenen Stilanweisungen (externe, im Dokument zentral oder im Tag vereinbarte Stile) wird ebenfalls nur die letzte Definition herangezogen.

Das W3-Konsortium hat bisher zwei CSS-Sprachversionen erarbeitet, eine dritte Version ist in der Diskussionsphase, die auf den Seiten des W3-Konsortiums eingesehen werden können:

- CSS-Version 1.0: http://www.w3.org/TR/REC-CSS1
- CSS-Version 2.0: http://www.w3.org/TR/REC-CSS2/
- CSS-Version 2.1: http://www.w3.org/TR/CSS21/

Vorliegende Anleitung beschreibt nur das prinzipielle Vorgehen; eine vollständige Erläuterung aller Stilvorgaben würden den Rahmen dieses Skripts bei Weitem sprengen. Die möglichen Stilanweisungen sind in einer separaten Tabelle notiert; eine vollständige Erläuterung erhalten Sie auch bei SELFHTML (http://de.selfhtml.org/).

2. Einbindung in HTML-Dateien

2.1. Verfahren der Einbindung von Stilvorlagen

Es gibt drei verschiedene Möglichkeiten der Angabe von Stilanweisungen:

1. Angabe direkt im HTML-Tag (*Inline Styles*),
2. Angabe von Stilangaben im Dokumentkopf (*Global Styles*),
3. Angabe von Stilangaben in separaten, zentralen Dateien (*External Styles*).

Die direkte Angabe im HTML-Tag ist einfach (**Inline Styles**): Fügen Sie dem zugehörigen Tag das Attribut `style="stilanweisungen"` hinzu und es wird ausgeführt. Können Sie die Anweisung keinem entsprechenden Tag zuweisen, z.B. für ein Wort in einem Absatz, so benutzen sich zum Umspannen das `<span>`-Tag, das ansonst keine weitere Funktion bewirkt.

Beispiel:

```
<p style="color: #00FF00">Text</p>
```

Der Text des Absatzes erscheint nun in grüner Farbe. Auf die Bedeutung der Anweisungen kommen wir später zu sprechen.

Im Zusammenhang mit der Verwendung von `style="stilanweisungen"` in Tags muss die Style Sheet-Sprache im Dokumentkopf mit einer `<meta>`-Anweisung ausdrücklich festgelegt werden (bei den anderen beiden Möglichkeiten ist die Festlegung bereits im Tag mit dem entsprechenden Attribut vorgesehen):

```
<head>
   ...
   <meta http-equiv="Content-Style-Type" content="text/css" />
   ...
</head>
```

Die zweite Möglichkeit, nämlich die Definition an zentraler Stelle (Global Styles), ist aber in der Regel sinnvoller. Die Nutzung derartiger Stilanweisungen wird im Dokumentkopf vereinbart, das zugehörige Tag ist `<style>`.

Beispiel:

```
<head>
   ...
   <style type="text/css">
      <!--
      h1, h2 { color: red; font-family: sans-serif }
      p { color: green; font-family: arial, sans-serif }
      // -->
   </style>
   ...
</head>
```

Die Kommentaranweisungen `<!-- ... //->` sind nur für Browser gedacht, die das `<style>`-Tag nicht kennen und den dazwischen stehenden Text auf dem Bildschirm ausgeben würden.

Bei den Angaben der Form:

```
   tag { stilanweisungen }
```

handelt es sich um die eigentlichen Stilvorlagendefinitionen: der Tag-Bezeichnung (ohne spitze Klammern) folgen die Stilangaben in geschweiften Klammern, die einzelnen Stilangaben sind untereinander durch Semikolons getrennt.

Drittens gibt es neben der Möglichkeit der Angabe aller Stilanweisungen im Dokumentkopf noch die Möglichkeit, alle oder ein Teil der Stilangaben in einer separaten Textdatei (External Styles), die dann üblicherweise die Dateierweiterung *.css erhält, zu notieren und diese aufzurufen. An Stelle der <style>-Definition wird folgende Zeile im Dokumentkopf eingetragen:

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

Achten Sie darauf, dass an Stelle des `<style>`-Tags das `<link>`-Tag benutzt werden muss. Die Angabe zum `href`-Attribut folgt denselben Regeln wie beim `<a>`-Tag.

Im Falle einer XHTML-Datei wird obige `<link>`-Anweisung nicht ausgewertet, so dass Sie für diesen Falle eine XML-spezifische Angabe nach folgenden Muster ergänzen müssen:

```
<?xml-stylesheet type="text/css" href="../../styles.css" ?>
<link rel="stylesheet" href="../../style.css" type="text/css" />
```

Die Style-Sheet-Datei enthält nur noch die eigentlichen Stilvorlagen, wie nachfolgendes Beispiel zeigt:

```
h1, h2 { color: red; font-family: sans-serif }
p { color: green; font-family: arial, sans-serif }
```

2.2. Style Sheet-Angaben für unterschiedliche Ausgabemedien

Es bedarf sich keiner größeren Erläuterung, dass der Wunsch bestehen könnte, dass Ausgaben zum Beispiel auf Drucker und Bildschirm mit unterschiedlichen Stildefinitionen erfolgen sollten. Dies ist in der CSS-Version 2.0 standardisiert.[1]

Dabei gibt es zwei Möglichkeiten:

1. Definition unterschiedlicher Style-Sheet-Bereiche mit der `@media`-Anweisung (im `<style>`-Tag oder in einer einzigen externen Datei,
2. Definition in verschiedenen Dateien.

Im ersten Falle sähe eine Definition wie folgt aus, unterschiedliche Bereiche werden über die Angabe `@media` vermittelt. Für verschiedene Ausgabegeräte gleichartig benutzte Anweisungen schreibt man ohne `@media`-Anweisung:

```
<head>
   ...
   <style type="text/css">
   <!--
   @media print
   {
     ... Style-Sheet-Angaben zum Drucken ...
   }
   @media screen, projection
   {
     ... Style-Sheet-Angaben zur Bildschirmausgabe ...
   }
   //-->
   </style>
   ...
</head>
```

Das Vorgehen ist in einer externen *.css-Datei sinngemäß möglich.

Die zweite Möglichkeit sieht die getrennte Definition von Stilvorlagen in getrennten Dateien für die unterschiedlichen Ausgabemedien vor: dazu werden `<link rel="stylesheet" ...>` bzw. für

[1] Wird von Internet Explorer 4.x, Netscape 6.0 und Mozilla unterstützt.

XHTML-Dateien `<?xml-stylesheet ...?>` für jedes Ausgabemedium gesondert notiert, zusätzlich muss das `media`-Attribut angegeben werden:

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

Derzeit sind nachfolgende `media`-Angaben standardisiert:

- `media="screen"` für Style Sheets, die bei der Bildschirmpräsentation wirksam sein sollen.
- `media="print"` für Style Sheets, die bei der Ausgabe über Drucker wirksam sein sollen.
- `media="aural"` für Style Sheets, die wirksam sein sollen, wenn der Inhalt der HTML-Datei per Sprachausgabe über Lautsprecher erfolgt.
- `media="projection"` für Style Sheets, die wirksam sein sollen, wenn der Inhalt der HTML-Datei per Diaprojektor oder Overhead-Projektor erfolgt.
- `media="braille"` für Style Sheets, die wirksam sein sollen, wenn der Inhalt der HTML-Datei über taktile Braille-Medien erfolgt.
- `media="tv"` für Style Sheets, die wirksam sein sollen, wenn der Inhalt der HTML-Datei über Fernsehtechnik erfolgt.
- `media="handheld"` für Style Sheets, die wirksam sein sollen, wenn der Inhalt der HTML-Datei über Handys, Palmtops oder ähnliche Geräte mit kleinem Display erfolgt.
- `media="all"` für Style Sheets, die in allen Medientypen wirksam sein sollen. Dies ist der Standard.

3. Formatvorlagen-Anweisungen

3.1. Definitionen mittels Selektoren

Elemente (d.h. Tag-Typen) oder Elementklassen werden über *Selektoren* beschrieben.

Für Tags können Sie, wie bereits angedeutet, Vorgaben in wie folgt definieren:

```
tag { liste_von_stilanweisungen }
```

Dem Tag-Namen (ohne spitze Klammern) folgt in geschweiften Klammern eine mit Semikolons getrennte Liste von Formatanweisungen. Jede einzelne Anweisung besteht aus einem Bezeichner, gefolgt von einem Doppelpunkt (kein Gleichheitszeichen!) und dem zugehörigen Wert; mehrere zugehörige Werte werden durch Kommata getrennt.

Beispiel:

```
<style type="text/css">
   <!--
   h1 { font-size: 48pt; color: #FF0000; font-style: italic }
   p,li  { font-size:12pt;
      line-height:14pt;
      font-family: Helvetica, Arial, "Times New Roman";
      letter-spacing: 0.2mm;
      word-spacing: 0.8mm }
   //-->
</style>
```

Wie Sie am Beispiel (`p,li`) erkennen, können Sie gleichartige Definitionen auch mehreren Tags gleichzeitig zuordnen.

Was Sie weiterhin erkennen, ist, dass Sie nur die Definitionen vornehmen müssen, die Sie gegenüber den Standardvorgaben ändern wollen. Begriffe, die aus mehreren Einzelwörtern wie im Falle von Schriften bestehen, werden in Anführungsstriche notiert.

Nun kann es vorkommen, dass Sie für ein und dasselbe Tag verschiedene Stilvorgaben definieren und benutzen möchten. Dies ist über die Definition von Klassen möglich: eine neue Tag-Stildefinition erfolgt von einem Punkt und einem Klassenbezeichner.

Beispiel:

```
<style type="text/css">
   <!--
   p { font-size: 10pt; color: black; }
   p.normal { font-size: 10pt; color: black; }
   p.gross { font-size: 12pt; color: black; }
   p.klein { font-size: 8pt; color: black; }
   .blau { color: blue; }
   //-->
</style>
```

Im obigen Beispiel sehen Sie, dass neben der Definition für den Absatz im Allgemeinen weitere drei Klassen definiert wurden: `normal`, `gross`, `klein`. Auf diese Klassen können Sie im Tag unter Nutzung des `class`-Attributes zugreifen, wie folgendes Beispiel zeigt:

```
<p class="gross">Ein Absatz in gro&szlig;er schwarzer Schrift</p>
```

Und – Sie müssen eine Stildefinition gar nicht an ein Tag binden, wie folgendes Beispiel zeigt:

```
<p class="blau">Ein Absatz in normaler blauer Schrift</p>
```

Zusätzlich zu einer „Tag-losen“ Unterklassendefinition können Sie auch ein unabhängiges Format über den `id`-Bezeichner vermitteln. Derartige `id`-Bezeichner werden im Style Sheet mit einem Doppelkreuz notiert:

```
<style type="text/css">
   <!--
   #fettkursiv { font-weight: bold; font-style: italic; }
   //-->
</style>
```

Der Bezug hierzu lautet wie folgt:

```
<p id="fettkursiv">Ein Absatz in normaler fetter und kursiver Schrift</p>
```

Natürlich bestehen Gemeinsamkeiten mit der Klassendefinition. Interessant ist aber, dass die Definition per `id`-Attribut die Kombination mit weiteren Formatangaben mittels `class`-Attribut ermöglicht. Beachten Sie allerdings, dass jeder Wert eines `id-Attributs` in der Datei nur einmal vorkommen sollte.

3.2. Definitionen mittels Kombinatoren

Elemente (d.h. Bezeichner für Tags), die in einer Abhängigkeit von anderen Elementen stehen, werden über *Kombinatoren* spezifiziert.

Nachfolgend zwei Beispiele:

Wenn Sie beabsichtigen, mehrere Tags zu schachteln und die Tag-Auszeichnung von eben dieser Schachtelung abhängig zu machen, so können Sie diese Definition mittels so genannter *descendant combinators* wie folgt vornehmen:

```
i { font-style: italic; color: #000000; }
h1 { color: #FF0000; }
h1 i { color: #0000FF; font-style: normal; }
```

Die Angabe `h1 i` bedeutet, dass für ein Konstrukt der Form `<h1> ... <i>...</i> ... </h1>` die hier angegeben Stilangabe für das `<i>`-Tag benutzt werden soll. Wird das `<i>`-Tag außerhalb von Überschriften des Typs `<h1>` benutzt, so erfolgt die Darstellung kursiv und in schwarz.

Eine gezieltere Definition dieser Art können Sie mit den *child combinators* (E > F; E * F) vornehmen (siehe nachfolgende Übersicht).

Als ein Beispiel für *adjacent operators*, die für aufeinander folgende Tags benutzt werden, soll die Absatzgestaltung mit Einrückung dienen. In vielen Publikationen werden Absätze eingerückt, in der Regel soll dies aber nicht für den Absatz gelten, der unmittelbar auf eine Überschrift folgt.

Die Formulierung hierfür könnte wie folgt aussehen:

```
<head>
 <title>Titel</title>
 <style type="text/css">

  p { text-indent: 20px; }
  h1 + p, h2 + p, h3 + p, h4 + p, h5 + p, h6 + p { text-indent: 0px; }

 </style>
</head>
```

3.3. Übersicht Selektoren und Kombinatoren

Der Sinn von Cascading Style Sheets beruht darauf, einem Element (HTML-Tag bzw. XML-Tag) bestimmte Attribute (z.B. Farbe, Rahmen etc.) zuzuweisen. Die Auswahl der Elemente erfolgt über *Selektoren*. Elemente, die in einer Abhängigkeit von anderen Elementen stehen, werden über *Kombinatoren* spezifiziert.

Nachfolgend finden Sie eine Übersicht der Selektoren, wie sie bereits im vorherigen Abschnitt behandelt wurden:

Art	Muster	Erläuterung	CSS
Universal selector	*	Repräsentiert jegliches HTML- bzw. XML-Element	**CSS2**
Type selectors	E	Repräsentiert jedes E-HTML- bzw. -XML-Element, z.B. h1	CSS1
Class selectors	E.myClass	Nur für HTML. Repräsentiert jedes E-Element, dessen "class"-Attribut den Wert "myClass" besitzt. Identisch mit E[class~="myClass"]. Dabei muss keine Element-Spezifikation erfolgen: .myClass ist identisch mit *[class~="myClass"].	CSS1
ID selectors	E#myId	Repräsentiert jedes E-Element, dessen "id"-Attribut den Wert "myId" besitzt. Identisch mit E[id~="myId"]. Dabei muss keine Element-Spezifikation erfolgen: #myId ist identisch mit *[id~="myId"].	CSS1

Die Definition von Formatvorlagen unter der Bedingung aufeinander folgender Tags wird durch zahlreiche Kombinatoren ermöglicht:

Art	Muster	Erläuterung	CSS
Descendant combinators (Contextual selector)	E F	Repräsentiert jedes F-Element, das vom E-Element umschlossen wird. h1 {color: red} em {color: red} h1 em {color: blue}	CSS1
Child combinators	E > F	Repräsentiert jedes F-Element, das ein Kindelement (child) des E-Elements ist. ol > li { ... } ul > li { ... }	**CSS2**
	E * F	Repräsentiert jedes F-Element, das ein Enkelkindelement (grandchild) bzw. Kindelement in einer tieferen Hierarchieebene des E-Elements ist.	**CSS3**
Direct Adjacent combinators	E + F	Repräsentiert jedes F-Element, das unmittelbar dem E-Element folgt. h1 + p {text-indent: 0px}	**CSS2**
Indirect Adjacent combinators	E ~ F	Repräsentiert jedes F-Element, das dem E-Element folgt.	**CSS3**
Attribute selectors	E[attribute]	Repräsentiert jedes E-Element, bei dem das Attribut "attribute" gesetzt ist, egal, welchen Wert es besitzt.	**CSS2**
	E[attribute="value"]	Repräsentiert jedes E-Element, dessen Attribut "attribute" genau den Wert "value" besitzt.	**CSS2**

Art	Muster	Erläuterung	CSS
	E[attribute~="value"]	Repräsentiert jedes E-Element, dessen Attribut "attribute"einen Wert besitzt, der genau einem der Listenelemente der Werteliste "value" entspricht. Die Listenelemente werden durch Leerraum getrennt.	**CSS2**
	E[lang|="en"]	Repräsentiert jedes E-Element, dessen "lang"-Attribut einen Bindestrich-geteilten Wert besitzt, der in unserem Falle (von links) mit "en" beginnt.	**CSS2**

3.4. Pseudoklassen und -elemente

Mit Selektoren bzw. Kombinatoren ist es nur möglich, sich auf Elemente (HTML-Tags bzw. XML-Tags) im Allgemeinen oder Hierarchien von Elementen zu beziehen. Für Spezifikationen über diesen Rahmen (Dokumentbaum bzw. Dokumentsprache [HTML, XML]) hinaus ermöglichen *Pseudo-Klassen* und *-Elemente* weitergehende Spezifikationen. Zum Beispiel erlaubt keine Dokumentsprache eine Spezifikation des Aussehens der ersten Zeile oder des ersten Buchstabens eines Absatzes oder zur Laufzeit generierte Inhalte einzufügen.

Pseudo-Klassen erlauben die Definition von Style Sheets für verschiedene Elementtypen, die sich durch Interaktion mit dem Dokument ergeben oder nicht aus dem Dokumentbaum erschlossen werden können.

Pseudo-Elemente erlauben die Definition von Style Sheets für Elementteile, auf die über die Dokumentsprache nicht zugegriffen werden kann. Dies betrifft z.B. den ersten Buchstaben oder die erste Zeile eines Absatzes.

Bei den Bezeichnern von Pseudo-Klassen und -Elementen wird nicht zwischen Groß- und Kleinschreibung unterschieden. Pseudo-Elemente müssen nach dem Element-Bezeichner notiert werden, dies ist bei Pseudo-Klassen nicht notwendig.

Beispiel:

```
<style type="text/css">
   <!--
   a:link { color: #FF0000; font-weight: bold }
   a:visited { color: #990000; }
   a:active { color: #0000FF; font-style: italic }
   a:hover { color: #0000FF; font-style: italic }
   // ->
</style>
```

Bei Style-Sheet-Definitionen von Pseudo-Elementen notieren Sie zuerst das betroffene HTML-Tag, im Beispiel das `<a>`-Tag für Verweise. Dahinter folgt ein Doppelpunkt und dahinter eine erlaubte Angabe, im Beispiel etwa `link` oder `visited`. Beachten Sie, dass dies keine frei wählbaren Namen sind, sondern feste Schlüsselwörter.

3.4.1. Übersicht Pseudo-Klassen

Wichtiges Beispiel:

Schlüsselwörter für das `<a>`-Tag:

- `link`: noch nicht besuchter Link.
- `visited`: bereits besuchter Link.
- `hover`: Link, über den sich der Mauszeiger befindet.
- `active`: Link mit gedrückter Maustaste.

Klasse	Erläuterung	CSS Level
Pseudoklasse Link: a:link a:visited	Repräsentiert einen noch nicht aufgesuchten bzw. einen bereits aufgesuchten Link. a:link { color: red } a:visited { color: blue }	CSS 1
Dynamische Pseudoklassen: E:active	Repräsentiert des Element E im aktivierten Zustand, d.h., der Anwender klickt das Element an und hält die Maustaste gedrückt.	CSS 1
E:hover	Repräsentiert des Element E in einem Zustand, in dem das Element mit der Maus überfahren, aber nicht angeklickt wird.	**CSS 2**
E:focus	Repräsentiert des Element E in einem Zustand, in dem es den Fokus besitzt.	**CSS 2**
Pseudoklasse language: E:lang()	Repräsentiert des Element E, das Text einer bestimmten Sprache enthält. Beispiel: Wahl unterschiedlicher typographischer Anführungszeichen.	**CSS 2**
Pseudoklasse Target: :target	Ziel-Element der diesbezüglichen URI.	**CSS 3**
Pseudoklasse Negation: :not()	Element, das nicht durch das Argument repräsentiert wird. a:not(:visited).	**CSS 3**

Klasse	Erläuterung	CSS Level
Pseudoklassen Nutzer-interface (z.B. Formu-lare): :enabled	Zustand, in dem Element freigegeben ist.	**CSS 3**
:disabled	Zustand, in dem Element nicht freigegeben ist.	**CSS 3**
:checked	Zustand, in dem Element ausgewählt ist.	**CSS 3**
:indeterminate	Element, dessen Zustand unbestimmt ist.	**CSS 3**
Strukturelle Pseudo-klassen: :root	Dokument-Wurzel. Im HTML ist dies das HTML-Element, in XML ist dies die zugehörige DTD, das Schema oder der Namenraum des XML-Dokuments.	**CSS 3**
:nth-child(an+b)	Element ist das jeweils *b*. Kindelement der Gruppen, die aus *a* Kindelementen bestehen. Mögliche Werte sind auch „odd“ und „even“. Ist a Null oder wird es weggelassen, so gibt es nur eine Gruppe. Die diesbezüglichen Kindelemente sind unterschiedlich, aber in derselben Hierar-chieebene. p:nth-child(2n+1) = p:nth-child(odd)	**CSS 3**
:first-child	Erstes Kindelement. :first-child ist identisch zu :nth-child(1).	**CSS 2**
:nth-last-child(an+b)	Element ist das jeweils *b*. Kindelement der Gruppen, die aus *a* Kindelementen bestehen, vom Hierarchieende aus gesehen. Mögliche Werte sind auch „odd“ und „even“. Ist a Null oder wird es weggelassen, so gibt es nur eine Gruppe. Die diesbezüglichen Kindelemente sind unterschiedlich, aber in derselben Hierarchie-ebene. p:nth-last-child(2n+1) = p:nth-last-child(odd)	**CSS 3**

Klasse	Erläuterung	CSS Level
:last-child	Letztes Kindelement. :last-child ist identisch zu :nth-last-child(1).	**CSS 3**
:nth-of-type(an+b)	Element ist das jeweils *b*. Kindelement der Gruppen aus gleichartigen Kindelementen derselben Hierarchieebene, die aus a Elementen bestehen. Mögliche Werte sind auch „odd“ und „even“. Ist a Null oder wird es weggelassen, so gibt es nur eine Gruppe.	**CSS 3**
:first-of-type	Erstes Element desselben Typs. :first-of-type ist identisch zu :nth-of-type(1).	**CSS 3**
:nth-last-of-type(an+b)	Element ist das jeweils *b*. Kindelement der Gruppen aus gleichartigen Kindelementen derselben Hierarchieebene, die aus *a* Elementen bestehen, vom Ende der Hierarchieebene aus gesehen. Mögliche Werte sind auch „odd“ und „even“. Ist *a* Null oder wird es weggelassen, so gibt es nur eine Gruppe.	**CSS 3**
:last-of-type	Letztes Element desselben Typs. :last-of-type ist identisch zu :nth-last-of-type(1).	**CSS 3**
:first-node	Erstes Knotenelement.	**Mozilla-Erweiterung**
:last-node	Letztes Knotenelement.	**Mozilla-Erweiterung**
:only-child	Element, das kein gleichrangiges Element (Sibling, Bruder bzw. Schwester) besitzt.	**CSS 3**
:only-of-type	Element, das kein gleichrangiges Element desselben Typs (Sibling, Bruder bzw. Schwester) besitzt.	**CSS 3**
:empty	Element, das kein Kindelement besitzt.	**CSS 3**
Pseudoklasse Content: :contains("substring")	Element, dessen Textinhalt die vorgegebene Zeichenkette „substring“ enthält.	**CSS 3**

3.4.2. Übersicht Pseudo-Elemente

Pseudo-Elemente sind am doppelten Doppelpunkt erkennbar. Die Angaben müssen aber auch bedient werden, wenn nur ein einfacher Doppelpunkt geschrieben wird. Da „::“ erst mit CSS 3 eingeführt wurde, geht man sicher, wenn nur „:“ angegeben wird.

Wichtige Beispiele:

Schlüsselwörter, einen Absatzes betreffend:

- `first-line`: Formatangabe für die erste Zeile eines Absatzes (wird von Netscape 4.x und Internet Explorer 4.x noch nicht interpretiert).
- `first-letter`: Formatangabe für das erste Zeichen der ersten Zeile eines Absatzes (wird von Netscape 4.x und Internet Explorer 4.x noch nicht interpretiert).

- `before`: Angabe (von variablen und typisierten) Text **vor** dem entsprechenden Absatztext. Beispiel:

```
<style type="text/css">
   <!--
   p.Hinweis:before { content: "Hinweis: " }
   // -->
</style>
```

- `after`: Angabe (von variablen und typisierten) Text **nach** dem entsprechenden Absatztext.

Die `content`-Angabe für die Pseudoformate ist sehr mächtig, so können enthalten sein:

- Text, wie oben im Beispiel gezeigt in Anführungszeichen gesetzt. Wenn Sie einen Zeilenumbruch benötigen, schreiben Sie an dessen Stelle die Escape-Sequenz „\A“.
- Inhalte aus anderen Quellen, Beispiel:
  ```
  @media aural {
     blockquote:after { content: url("beautiful-music.wav") }
  }
  ```
  Das Beispiel würde bewirken, dass am Ende eines Zitats die genannte Sound-Datei abgespielt würde.
- `attr`: Ausgabe des Textes von Attributen, Beispiel:
  ```
  img:before { content: attr(alt) }
  ```
  Das Beispiel würde bewirken, dass vor dem Bild der alternative Text angegeben würde. Er wäre auch dann sichtbar, wenn das Bild nicht angezeigt werden kann.
- Mit `open-quote` und `close-quote` lassen sich ein öffnendes und schließendes Anführungszeichen angeben. Diese berücksichtigen die Schachtelungstiefe von Anführungszeichen.

- Definition von Zählern für Absatznummerierung (siehe S. 34). Die Berücksichtigung von Zählern wird bisher nur vom Browser Opera vorgenommen.

Element	Erläuterung	CSS Level
::first-line	Betrifft die Formatierung der ersten Textzeile eines Elements.	CSS 1
::first-letter	Betrifft die Formatierung des ersten Buchstabens eines Elements.	CSS 1
::before	Text, der dem im Dokument angegebenen Element-Text vorangestellt wird. Dieser Text kann Inhalte enthalten, die erst zur Laufzeit generiert werden, z.B. Abschnitts- oder Gliederungsnummerierung.	**CSS 2**
::after	Text, der dem im Dokument angegebenen Element-Text nachgestellt wird. Dieser Text kann Inhalte enthalten, die erst zur Laufzeit generiert werden, z.B. Abschnitts- oder Gliederungsnummerierung.	**CSS 2**
::selection	Betrifft die Formatierung des ausgewählten Elements. Kann zusammen verwendet werden mit :checked.	**CSS 3**

4. Kaskade und Vererbung

Ein bedeutender Problemkreis stellt die Frage dar, was passiert, wenn Eigenschaften nicht oder mehrfach in den Stilvorlagen definiert werden. Das Problem wird mit zwei Mechanismen vollständig gelöst: mit Vererbung bei fehlenden Angaben und mit der Kaskade im Falle von Mehrfachangaben.

4.1. Reihenfolge bei der Bearbeitung der Kaskade

Die Bearbeitung der Kaskade erfolgt in vier Schritten:

1. Vorab werden alle diejenigen Style-Sheet-Deklarationen ermittelt, die für den gewünschten Medientyp zutreffend sind.
2. Im ersten Schritt erfolgt eine Sortierung der Deklarationen nach Wichtung (*normal* oder *wichtig* (*!important*)) und nach Ursprung (Autor, Nutzer und Anwenderprogramm (zumeist ist dies ein Browser)). In absteigender Reihenfolge sind dies:
   1. Die !important-Deklarationen des Nutzers,
   2. die !important-Deklarationen des Autors,
   3. die normal-gewichteten Deklarationen des Autors,
   4. die normal-gewichteten Deklarationen des Nutzers,
   5. die Deklarationen des Anwenderprogramms.
3. Im zweiten Schritt werden Regeln gleicher Wichtung und gleichen Urpsrungs nach ihrer spezifische Wirksamkeit (*engl.* specifity) sortiert und hierbei die Deklarationen höchster spezifischer Wirksamkeit ausgewählt (siehe unten).
4. Im dritten Schritt werden die Regeln gleicher Wichtung, gleichen Ursprungs und gleicher spezifischer Wirksamkeit sortiert: Die letztgenannte Regel wird ausgewählt.

Dazu noch einige Anmerkungen:

1. !important-Regeln dienen dem Interessenausgleich zwischen dem Autor und dem Nutzer. Üblicherweise sollte dabei die Stilvorlage des Autors das gleichgewichtete des Benutzers ersetzen. Im Gegensatz zur Spezifikation CSS Level 1 besitzen nun aber die !important-Deklarationen des Nutzers die höchste Priorität, damit der Nutzer seinem Bedarf entsprechend eingreifen kann (z.B. für größere Schriften oder spezielle Farben-Kombinationen).
2. Bei zusammenfassenden Eigenschaften (*engl.* shorthand property) wie z.B. `background` gilt, dass eine vorgenommene !important-Regel auch für alle untergeordneten Eigenschaften gilt.
3. Importierte Regeln (@import) besitzen dieselbe Wirkung wie die Stilvorlage, in die sie importiert wurden.

4.2. Spezifische Wirksamkeit von Selektoren

Um die spezifische Wirksamkeit von Selektoren zu bestimmen, wird die Anzahl von vier Selektorbestandteile ermittelt:

1. Wert *a* = 1, wenn das `style`-Attribut in einem Tag verwendet wird.
2. *b* ist die Summe der in der Regel verwendeten `id`-Selektoren.
3. *c* ist die Summe der in der Regel verwendeten Klassen- und Pseudoklassen-Selektoren.
4. *d* ist die Summe der restlichen Element- und Pseudo-Element-Selektoren außer dem *-Operator.

Diejenige Regel besitzt die höchste spezifische Wirksamkeit, die den höchsten Wert *a* besitzt, bei Gleichheit zählt in absteigender Reihung der jeweils höchste Wert *b*, *c* oder *d*.

Beispiele

Regel	Spezifische Wirksamkeit			
	a	b	c	d
style = ""	1	0	0	0
#container ul li	0	1	0	2
#container ol	0	1	0	1
li.red.level	0	0	2	1
.red.level	0	0	2	0
a:link	0	0	1	1
ul[id="nav"]	0	0	1	1
h1 + p	0	0	0	2
li:first-line	0	0	0	2
*	0	0	0	0

4.3. Vererbung

Besitzt ein Element bestimmte Stilregel nicht, so erbt es sie von seinem übergeordneten Element bzw. dessen übergeordnetem Element(en). Die Vererbung wird standardmäßig ausgeführt; nur in Ausnahmefällen muss man sie mit dem Wert `inherit` erzwingen.

Im folgenden Beipsiel:

```
<h1>Dies ist eine <em>wichtige</em> Überschrift</h1>
```

übernimmt das `em`-Tag die Schriftgröße der Überschrift `h1`, weil im Fall des `em`-Tags nur dessen Schriftstil geändert wird.

4.4. Referenzen

- Assigning property values, Cascading, and Inheritance (CSS 2.1): http://www.w3.org/TR/CSS21/cascade.html

5. Ausgewählte Stilangaben

5.1. Maßeinheiten und Farbangaben für Style Sheets

5.1.1. Maßeinheiten

Bei allen numerischen Angaben innerhalb von CSS-Style-Sheets, also etwa bei Schriftgrößen, Absatzabständen oder Rändern, stehen Ihnen alle weit verbreiteten Maßeinheiten zur Verfügung. Dabei sind absolute Angaben (z.B. Millimeter) und relative Angaben (z.B. Prozent gegenüber „normal“) möglich. Nachfolgende Abkürzungen stehen Ihnen hierbei zur Verfügung:

Absolute Angaben (weniger für Bildschirm geeignet):

- pt für Punkt (= 1/72 Zoll).
- pc für Pica (= 12 Punkt)
- in für Inch (= 2,54 cm)
- mm für Millimeter
- cm für Zentimeter.

Relative Angaben:

- em für „bezogen auf elementeigene Schrifthöhe“, Breite des Kleinbuchstaben „m“.
- ex für „bezogen auf elementeigene Höhe des Buchstabens x“.
- px für Pixel (Bildschirmpunkt: auf dem Bildschirm eigentlich eine Absolutangabe).
- % für Prozent gegenüber Elementnorm.

Benutzen Sie bei numerischen Bruchzahlen stets den Punkt als Dezimalzeichen, nicht das deutsche Komma, also etwa 0.5 cm und nicht 0,5 cm.

Ein wichtiges Problem stellt die Nutzung der beliebten Maßeinheiten pt und px dar:

- **pt** ist auf dem Papier eine Absolutangabe; auf dem Bildschirm hängt die Größe von der Bildschirmauflösung und der Monitorgröße ab. pt ist für Ausgaben auf Papier ideal.
- **px** ist auf dem Bildschirm eine Absolutangabe; auf dem Papier ist die Maßangabe unbestimmt. pt ist für „pixelgenaue“ Ausgaben auf dem Bildschirm ideal.

Die Nutzung unterschiedlicher `@media`-Anweisungen (siehe oben) ermöglicht die optimale Maßangabe auch für verschiedene Ausgabemedien.

5.1.2. Farbangaben

Prinzipiell gibt es nichts Neues für die Art und Weise der Angabe von Vorder- und Hintergrundfarben: sie folgen den Regeln der Farbangaben für die Verwendung von Tag-Attributen: entweder in Form #RRGGBB oder mit einem der definierten Farbnamen. Eine weitere Möglichkeit besteht in der Nutzung des Funktionsausdruckes `rgb(rrr, ggg, bbb)`, der entweder die dezimale Angabe (0–255) oder eine prozentuale Angabe (0 %–100 %) der Intensitäten der drei Farbauszüge Rot, Grün, Blau ermöglicht.

Nachfolgender Style Sheet-Auszug zeigt Beispiele auf:

```
<style type="text/css">
   <!--
   body { color: #000000; background-color: white; }
   p { color: rgb(64, 64, 64); backgrund-color: rgb(80%, 80%, 80%); }
   // -->
</style>
```

5.2. Wichtige Schriftauszeichnungen

Mit `font-family` können Sie verschiedene, auch mehrere Schriftartenfamilien wie Arial, Helvetica, Times New Roman definieren.

Beachten Sie hierbei folgendes:

- Schriftarten, die nicht angezeigt werden können, werden ignoriert und wenn die Zugehörigkeit erkennbar ist, durch eine ähnliche Schriftart ersetzt. Man kann auf den Auswahlprozess selbst Einfluss nehmen, indem man mehrere Schriftarten vorgibt.
- Schriftarten aus mehreren Teilwörtern werden in Anführungszeichen gesetzt, Beispiel: `font-family: "Times New Roman"`.
- Die meisten Schriftarten sind betriebssystemspezifisch: Arial gibt es nur unter Microsoft ® Windows, dafür gibt es Helvetica nur auf Macs. Gute Style Sheets benennen mehrere mögliche Schriften in der gewünschten (hierarchischen) Reihenfolge.
- Um sicher zu gehen, fügen Sie zum Schluss einen generischen Schriftfamiliennamen an: `serif`: Schrift mit Serifen (z.B. Times), `sans-serif`: serifenlose Schrift (z.B. Arial), `monospace`: dicktengleiche Schrift (z.B. `Courier`). Weiter sind möglich: `cursive`: kursive Schrift, `fantasy`: eine fantasievolle Schrift.

Schriftgröße, Schriftgewicht, Schriftstil und Schriftvarianten können über `font-size`, `font-weight`, `font-style` und `font-variant` bestimmt werden.

Neben Maßangaben für Schriftgrößen `font-size` wie 12 pt lassen sich auch folgende Relativangaben bezogen auf die Normalgröße vorgeben (Als Richtwert dient, dass die jeweils größere oder kleinere Angabe um 50 % größer oder kleiner als ihr Vorgänger ist):[2]

- `xx-small`: winzig (12,5 % – 25 %).
- `x-small`: sehr klein (25 % – 50 %).
- `small`: klein (50 % – 75 %).
- `medium`: mittel (100 %).
- `large`: groß (150 %).
- `x-large`: sehr groß (200 % – 225 %).
- `xx-large`: riesig (300 % – 350 %).

- `smaller`: sichtbar kleiner als normal; 50 % kleiner als der Zeichensatz des übergeordneten Elements.
- `larger`: sichtbar größer als normal; 50 % größer als der Zeichensatz des übergeordneten Elements..

Das Schriftgewicht `font-weight` wird üblicherweise in Promille angegeben. Folgende Angaben sind möglich:

- 100 (extra-dünn), 200, 300, 400, 500, 600, 700, 800, 900 (extra-fett).
- `bolder`: extra-fett (900).
- `bold`: fett (700).
- `normal`: normal, medium (500)
- `light`: dünn (300).
- `lighter`: extra-dünn (100).

Folgende Angaben sind für den Schriftstil `font-style` möglich:

- `italic`: Schriftstil kursiv.
- `normal`: Schriftstil normal (aufrecht).
- `oblique`: Schriftstil (ebenfalls) kursiv.

[2] Die Werte xx-small, x-small, small, medium, large, x-large und xx-large entsprechen die Größenangaben 1–7 des proprietären <font size>-Tags.

Folgende Angaben sind für die Schriftvariante `font-variant` möglich:

- `normal`: normale Schriftvariante.
- `small-caps`: Kapitälchen.

Wie nachfolgendes Beispiel zweigt, lassen sich o.g. Angaben auch mit `font` zusammenfassen:

```
<style type="text/css">
   <!--
   p.a { font-size: 12pt; font-weight: bold; font-style: italic }
   p.b { font: Arial 12pt bold italic small-caps }
   // -->
</style>
```

Weiterhin lassen sich für Text Dekorationen und Transformationen angeben.

Folgende Angaben für Text-Dekoration `text-decoration` sind möglich:

- `none`: normal (keine spezielle Text-Dekoration).
- `blink`: blinkend.
- underline: unterstrichen.
- `overline`: überstrichen.
- `line-through`: durchgestrichen.

Folgende Angaben für Text-Transformation `text-transformation` sind möglich:

- `none`: normal (keine spezielle Text-Transformation).
- `capitalize`: Wortanfänge in Großbuchstaben.
- `uppercase`: nur Großbuchstaben.
- `lowercase`: nur Kleinbuchstaben.

Zeichen-, Wort- und Zeilenabstände lassen sich mit `letter-spacing`, `word-spacing` bzw. `line-height` vereinbaren.

Ausrichten lässt sich der Text mit `text-align` (auch `alignment`; horizontal) und `vertical-align` (vertikal).

Folgende Werte sind für `text-align` möglich:

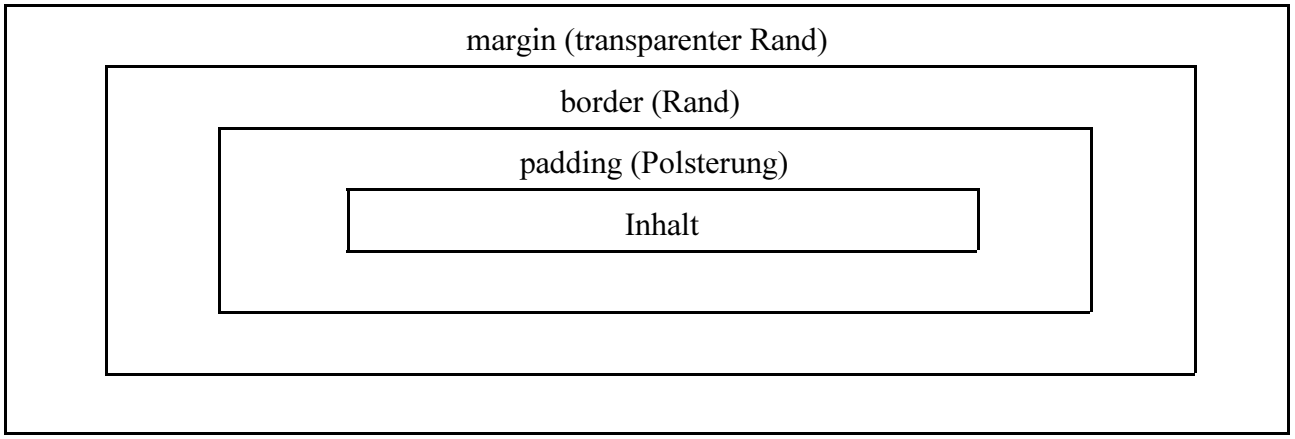
- `center`: mittig ausrichten.

Folgende Werte sind für `vertical-align` möglich:

- `justify`: als Blocksatz ausrichten.
- `left`: linksbündig ausrichten (Voreinstellung).
- `right`: rechtsbündig ausrichten.
- `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.

5.3. Das Boxenmodell

Mit der CSS-Version 1.0 wurde ein einfaches Kasten-orientiertes Formatiermodell eingeführt, dass es jedem darstellbaren Tag-Element erlaubt, sich mit rechteckigen Kästen zu umgeben. Die Breiten der Rahmen sind auf allen vier Seiten (`top`, left, bottom, right) individuell einstellbar.



Beispiele für Randvorgaben:

```
<style type="text/css">
   <!--
   p.a { margin-top: 10px; margin-left: 5px; margin-bottom: 10px;
      margin-right: 5px }
   p.b { margin: 10px 5px 10px 5px }
```

```
   // -->
</style>
```

Die Angaben für `p.a` und `p.b` sind identisch. Fehlende Angaben bei `margin` werden vom vorangegangenen Maß übernommen, z.B. bedeutet `margin: 10 px`, dass alle Ränder 10 px breit sind.

Der Hintergrund kann mit einer Farbe oder einem Bild gefüllt werden, das Hintergrund kann frei positioniert werden und einzeln stehen oder wiederholt werden. Beispiel:

```
<style type="text">
   <!--
   body { background: url(back.gif) repeat-y left;
          background-color: yellow }
   // -->
</style>
```

5.4. Positionierung von Elementen

Sie können (prinzipiell) für alle Elemente die Positionierungsart, die Startposition und die Größe angeben.

Beispiel:

```
<div style="position: absolute; top: 50px; left: 50px; width: 400px;
   height: 100px">Textinhalt</div>
```

Folgende Werte für `position` sind möglich:

- absolute: absolute Positionierung, gemessen vom ursprünglichen Fensterrand, Element wird beim Blättern mit gerollt.
- fixed: absolute Positionierung, gemessen vom Fensterrand, bleibt beim Scrollen stehen.
- relative: relative Positionierung, gemessen zum Vorgänger-Element.
- static: keine spezielle Positionierung, normaler Elementfluss (Vorgabewert).

Anhand der vorgestellten Werte erkennen Sie, dass hiermit der gewohnte Textfluss aufgebrochen werden kann, z.B. können sich Elemente überlappen. Die Angabe `position` muss mit einem Startwert kombiniert werden, sonst macht sie keinen Sinn. Als Startwerte kommen entweder `top` oder `bottom` bzw. `left` oder `right` in Frage.

Obwohl die Positionierungsangaben eigentlich bei allen Tags anwendbar sein sollten, geht man momentan sicher, wenn man diese Positionierungsangaben ausschließlich auf das `<div>`-Tag anwendet.

5.5. Mehrspaltiger Text

Als Vorgriff auf CSS-Version 3.0, aber bisher von keinem Browser unterstützt, sei hier die Möglichkeit der Realisierung mehrspaltigen Textflusses genannt. Es lässt sich die Spaltenanzahl `column-count`, der Spaltenabstand `column-gap`, Typ, Dicke und Farbe des Trennstriches `column-rule` definieren.

Beispiel:

```
<div style="column-count: 2; column-gap: 10px; column-rule: 1px solid black">
   Textinhalt ... Ende</div>
```

Für das Aussehen des Trennstriches können Sie u.a. folgende Werte wählen:

- `none`: keine Trennstriche (Normaleinstellung).
- `dotted`: gepunktete Trennstriche.
- `dashed`: gestrichelte Trennstriche.
- `solid`: durchgezogene Trennstriche.
- `double`: doppelt durchgezogene Trennstriche.

5.6. Absatz- und Zeichenbezogenheit

Normalerweise braucht man sich keine Gedanken zu machen über Absatz- und Zeichenbezogenheit eines Elementes zu machen; dies ist in den Vorgabeeinstellungen bereits geregelt. Braucht man weitere absatz- oder zeichenbezogene Stildefinitionen, so kann man sie vom `<div>`- bzw. `<span>`-Tag ableiten.

Wenn man die Angaben nun doch ändern möchte, so ist dies mit `display` möglich. U.a. sind folgende Werte für die Anzeige möglich:

- `block`: Element ist absatzbezogen.
- `inline`: Element ist zeichenbezogen.
- `none`: Element wird nicht angezeigt.

Interessant wird dies jedoch, wenn Sie mit Hilfe von Tabellen ein „pixelgenaues“ Design unter Einbezug von Bildern in XHTML (XML) erreichen wollen. Bilder sind (natürlich) `inline`-Elemente, die dargestellte Zeilenhöhe wird vom größeren Wert Zeilenhöhe oder Bildhöhe bestimmt: Bei Bildern, die eine geringere Höhe als die Zeilenhöhe besitzen, wird auch dann die Zeilenhöhe eingestellt, wenn gar kein Text angegeben wurde. In diesem Fall müssen Sie das Bild zum absatzbezogenen Element machen.

Dies soll am Beispiel einer Horizontallinie demonstriert werden:

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

5.7. Automatische Absatznummerierung

Absatznummerierung ist im Prinzip nichts Neues: seit der ersten Version von HTML ist sie für nummerierte Listen eingeführt. Leider nicht für andere Formen von Absätzen. Wie bereits angedeutet, ist dies in der Zukunft aber mit Hilfe der Pseudoformate `:before` und `:after` möglich.

Hierzu müssen Sie über `content` definieren:

- Zähler z.B. mit der Funktion counter(), sie hat zwei Formen: entweder `counter(name)` oder `counter(name, style)`. Die Standardvorgabe für style ist “decimal”.
- Der (die) Zähler können mit `counter-increment` und `counter-reset` gesteuert werden, als Wert erhalten sie den Namen des Zählers, dessen Wert erhöht oder rückgesetzt werden soll.

Nachfolgendes Beispiel definiert die Absatznummerierung am Beispiel der Überschriften `<h1>` und `<h2>`, die in der Form „Kapitel 1.“, „1.1“ usw. erscheinen sollen:[3]

```
h1:before {
    content: "Kapitel " counter(kapitel) ". ";
    counter-increment: kapitel;     /* Addiere 1 zu kapitel */
    counter-reset: unterkapitel;    /* Setze unterkapitel auf 0 */  }
h2:before {
```

[3] Wenn counter() in einem Browser nicht implementiert ist, so wird (oder sollte sinnvollerweise) auch der restliche Text der content-Angabe nicht dargestellt werden.

```
    content: counter(kapitel) "." counter(unterkapitel) " ";
    counter-increment: unterkapitel; }
```

Bisher zeigt nur Opera 5 und 6 die Absatznummerierung richtig an ...

6. Ausgabe auf Drucker

Neben der Bildschirmausgabe spielt die Ausgabemöglichkeit auf einem Drucker eine große Rolle. Der wesentlichste Unterschied ergibt daraus, dass die Druckausgabe seitenweise erfolgt. Hierzu werden weitere Eigenschaften benötigt: im Wesentlichen betreffen sie das Papierformat und das Verhalten beim Seitenumbruch.

Nachfolgende Tabelle zeigt ausgewählte Eigenschaften auf:

Eigenschaften	Werte	Beispiele Bemerkungen	CSS Level
size	(länge){1,2}, auto, portrait, landscape, inherit	Ausrichtung und Orientierung einer Seite. @page { size: 8.5in 11in; }	**CSS 2**
orphans	Anzahl Zeilen	Minimale Zeilenanzahl eines Absatzes am Seitenende (Hurensohn).	**CSS 2**
widows	Anzahl Zeilen	Minimale Zeilenanzahl eines Absatzendes am Seitenanfang (Schusterjunge).	**CSS 2**
page-break-after	auto, always, avoid, left, right, inherit	Verhalten Seitenumbruch nach Element.	**CSS 2**
page-break-before	auto, always, avoid, left, right, inherit	Verhalten Seitenumbruch vor Element.	**CSS 2**
page-break-inside	auto, avoid, inherit	Verhalten Seitenumbruch innerhalb des Elements.	**CSS 2**

Auf die Angabe einer Seitengröße sollte, von wenigen Ausnahmen abgesehen, verzichtet werden, hier sollte auf die Standardvorgaben des Nutzers zurückgegriffen werden; ansonst ist die Angabe von Papierbreite und -länge möglich. Sinnvoll ist aber die Vorgabe des Formats (Hoch- bzw. Querformat).

Sinnvoller ist aber die Definition von Seitenrändern.

Farbige Bildschirmausgaben sollten auf die üblicherweise vorhandene Schwarz-Weiß-Ausgabe eines Druckers angepasst werden; dies ist insbesondere bei hellen Schriftfarben notwendig.

Was auf dem Bildschirm keine Rolle spielt, bekommt bei der Ausgabe auf Papier umso höhere Bedeutung:

- Überschriften sollten nicht ohne Absatz am Seitenende stehen, weiterhin darf eine Überschrift am Seitenende nicht umgebrochen werden,
- Absätze sollten am Seitenende mindestens zwei Zeilen und am Seitenanfang ebenfalls mindestens zwei Zeilen besitzen (diese typographischen Kardinalfehler heißen Hurenkind (orphan) bzw. Schusterjunge (widow)).
- Innerhalb eines vorformatierten Textes (<pre>), wie er z.B. für Quelltexte benutzt wird, sollte kein Seitenumbruch stattfinden.
- Eine Liste sollte nicht ohne einleitenden Text auf der neuen Seite stehen.

Nachfolgend ein Auszug aus einer CSS-Datei zur Untermauerung:

```
body { ... }

@media print {
  @page          { size: portrait; margin: 2cm; }
  body           { background: #FFFFFF; border: 0px; }
  hr             { color: #000000; background: #000000; }
  p, li          { color: #000000; }
  h1, h2, h3,
  h4, h5, h6     { page-break-after: avoid; page-break-inside: avoid;
                   color: #000000; }
  blockquote,
  pre            { page-break-inside: avoid; }
  ul, ol, dl     { page-break-before: avoid; }
}
```

7. Problemfälle

7.1. Darstellung von Tabellen unter HTML 4.0 und XHTML 1.0

Im Rahen der Neustandardisierung von HTML 4.0 und XHTML 1.0 wurde insbesondere die Darstellung von Tabellen verändert. Bemerkbar wird dies erst, wenn man einen Browser benutzt, der diese Standards unterstützt.

7.1.1. Absatzränder in Tabellenzellen

HTML-4.0- und XHTML-1.0-Dokumente stellen Ränder von Absätzen und Überschriften im Dokument und in Tabellenzellen grundsätzlich dar. Störend ist häufig, dass der obere Rand des ersten bzw. der untere Rand des letzten Absatzes ebenfalls dargestellt werden.

Dieses Problem lässt sich mit den Pseudoformaten first-child und last-child lösen.

Beispiel für eine Problemlösung mittels Cascading Style Sheets:

```
body > *:first-child, td > *:first-child, th > *:first-child
  { margin-top: 0px; }
body > *:last-child,  td > *:last-child, th > *:last-child
  { margin-bottom: 0px; }
```

7.1.2. Höhe von Tabellenzellen

HTML-4.0- und XHTML-1.0-Dokumente stellen standardmäßig Tabellenzellen mit Höhen als Vielfaches von Textzeilenhöhen dar. Dies betrifft auch Bilder, die den Wert „inline“ der CSS-Eigenschaft „display“ besitzen. Soll die Höhe einer Tabellenzelle von der Bildhöhe abhängen, so muss der CSS-Eigenschaft „display“ der Wert „block“ zugewiesen werden.

Beispiel für eine Problemlösung mittels Cascading Style Sheets:

```
<table summary="Beispiel f&uuml;r Horizontallinie" bgcolor="#961E00"
  width="80%" cellspacing="0" cellpadding="0" border="0" align="center">
<tr>
<td><img src="dot.gif" width="1" height="1" alt=""
  style="display: block" /></td>
</tr>
</table>
```

7.2. Unterschiedliche Schriftgröße auf Windows- und Macintosh-Bildschirmen

In Punkt (pt) angegebene Schriftgrößen erscheinen auf Windows- und Macintosh-Rechnern in deutlich unterschiedlicher Größe, die Darstellung auf Macintosh-Rechnern ist kleiner. Die Ursache liegt darin, dass Windows-Systeme die Umrechnung von Papierlängenmaßen auf Bildschirmpunkte (Pixel, px) mit 96 dpi (dots per inch), Macintosh-Systeme dagegen mit 72 dpi vornehmen. Die Ausgabe auf Windows-Rechnern ist daher um (96/72 – 1) × 100 % = 33 % größer.

Man kommt letztendlich nicht umhin, für beide Plattformen verschiedene Formatvorlagen zu schreiben, die sich nur in der Schriftgröße unterscheiden. Der Austausch kann durch ein einfaches JavaScript-Skript oder eine Einfügung seitens des Servers erfolgen.

Beispiellösung mittels JavaScript:

```
<head>
 <title>Titel</title>
 ...
 <link rel="stylesheet" type="text/css" href="mac.css"
  title="Main Style Sheet" />

 <script language="JavaScript" type="text/javascript">
  if ((navigator.appVersion.indexOf("Win") >= 0)) {
   document.write("<link rel=\"stylesheet\" href=\"win.css\"
   title=\"Main Style Sheet\" \/>"); }
 </script>
</head>
```

Im Falle von Windows wird eine zweite Stilvorlage nachgeladen, die im Sinne der Kaskadierung (CSS) die erste Definition vollständig überschreibt.

Lösung mittels PHP (das Skript wird in das Dokument eingetragen, dass dann natürlich die Dateierweiterung php tragen muss. Der Server muss PHP unterstützen.):

```
<?php
if (eregi('Win',$HTTP_USER_AGENT))
 echo "<link rel=\"stylesheet\" href=\"IhrWinURL\" type=\"text/css\">\n";
else
 echo "<link rel=\"stylesheet\" href=\"IhrMacURL\" type=\"text/css\">\n";
?>
```

Die Variable `$HTTP_USER_AGENT` ist im PHP vordefiniert und enthält Angaben zum Betriebssystem, unter dem der eingesetzte Browser läuft. Die Funktion eregi() sucht nach einem Zeichenmuster in der angegebenen Zeichenkette ohne Berücksichtigung von Groß- und Kleinschreibung.

Wird die Zeichenkette „Win“ gefunden, wird die erste Zeichenkette in das Dokument eingeschrieben, ansonst die zweite.

Natürlich lässt sich Derartiges auch mit ASP (Advanced Server Pages, Skriptsprache von Microsoft) bzw. mit Server-Side-Includes bewerkstelligen.

8. Anhang 1: Maßeinheiten und Begriffe

8.1. Maßeinheiten

8.1.1. Computermaßeinheiten

Pixel

Eine Maßeinheit insbesondere für den Einsatz an Bildschirmen: Es entspricht der Höhe bzw. Breite eines Bildschirmpunktes.

8.1.2. Maßeinheiten in der Typographie

Cicero

Eine hauptsächlich in Europa verwendete Einheit im Didot-Maßsystem: Ein Cicero besteht aus 12 Didots, d.h. etwas größer als ein Pica.

Didot

Typographische Maßeinheit: 1 Didot = 0,376 mm.

Pica

12 Punkt bilden ein Pica, sechs Pica ein Zoll.

Punkt

Der Punkt ist eine typographische Maßeinheit und entspricht bei DTP-Systemen 1/72 Zoll = 0,35277 mm. Vergleichbare Maßeinheiten stellen in deutschsprachigen Ländern das Didot (=0,376 mm) und in anglo-amerikanischen Ländern das Pica-Point (=0,351 mm dar).

8.2. Schriften

Antiqua

Alle runden Schriften mit Ausnahme der Schreibschriften.

Dickte

Bezeichnet den Raum, den ein Buchstabe zwischen seinen Nachbarn beansprucht, einschließlich des äußeren Leerraums.

Durchschuss

Dieser Begriff aus der Bleisatz-Zeit benennt den Zwischenraum zweier Zeilen.

Font

Mit diesem Begriff ist eine Schriftart gemeint, er bezeichnet aber auch den kompletten Zeichensatz einer Schrift in einer Größe.

Fraktur

Deutsche runde Schrift.

Gemeine

Kleinbuchstaben einer Schrift.

Geviert

Bezeichnet im Satz einen festen Zwischenraum auf der Basis eines Quadrats, dessen Seitenlängen der Kegelhöhe der laufenden Schrift entspricht. Das Geviert (auch Halb- und Viertelgeviert) ist schriftabhängig. Man verwendet es zur Absatzmarkierung und zum Hervorheben einzelner Wörter.

Grotesk

So nannte man im 19. Jahrhundert bei deren Aufkommen die serifenlosen Antiqua-Schriften.

Halbgeviert

Ein Leerraum, der halb so breit, wie ein Geviert ist.

Hurenkind, Hurensohn (siehe auch Schusterjunge)

Die letzte Zeile eines Absatzes, die oben auf der nächsten Seite gedruckt wird. Satztechnischer Kardinalfehler.

Initial

Großbuchstabe am Anfang eines Kapitels oder Absatzes. Häufig mit größerem Schriftgrad und Ornamenten oder Bildmotiven ausgeschmückt.

Interpunktionsraum

Ein Interpunktionsraum entspricht der Breite eines Punkts einer Schrift.

Kapitälchen

Auszeichnungsschrift aus kleinen Großbuchstaben mit der Höhe der Mittellinie (entspricht der Höhe des kleinen x).

Ligaturen

Doppelbuchstaben wie ff und ss oder Kombinationen wie fl oder ft, die wie ein Zeichen behandelt werden. Auch das deutsche ß ist genau genommen eine Ligatur: aus s und z.

Oberlänge

Teil eines Kleinbuchstabens, der über den Buchstabenkörper hinausragt. Wie z.B. b, d, f...

Schusterjunge (siehe auch Hurenkind)

Erste Zeile eines Absatzes am Ende einer Textspalte. Satztechnischer Kardinalfehler.

Serifen

Teil von Buchstaben: Waagerechte Striche an den Enden der nach oben ragenden bzw. der nach unten führenden senkrechten Linien im Buchstaben. Ein typische Serifen-Schrift ist Times, eine typische serifenlose Schrift ist Arial bzw. Helvitica. Serifen-Schriften gelten gemeinhin als schöner, serifenlose Schriften lassen sich besser lesen, insbesondere für Menschen mit Sehschwächen.

Sperren

Eine Möglichkeit der Schriftauszeichnung, bei der die Abstände der Buchstaben geringfügig und gleichmäßig vergrößert werden.

Unterlänge

Teil eines Kleinbuchstabens, der unter der Grundlinie liegt. z.B. g, j, q, y.

Unterschneidung

Die Ausrichtung des Raumes zwischen Zeichenpaaren.

Unterschneidungspaar

Besteht aus zwei Zeichen. Der Abstand zwischen den Zeichen wird durch den Unterscheidungswert bestimmt.

Unterschneidungswert

Legt den Abstand zwischen einem Zeichenpaar fest.

Versalien

Großbuchstaben einer Schrift.

x-Höhe

Die Höhe eines kleingeschriebenen x einer Schrift, gemessen von der Grundlinie.

8.3. Standards

ASCII

American Standard Code for Information Interchange. Einfaches Format zum Speichern von Texten und Vorschrift für die Belegung der Tastatur.

PDF

Portable Document Format. Weiterentwicklung des PostScript-Formats durch Adobe.

PostScript

Spezielle Programmiersprache zur Beschreibung von grafischen Objekten und Schriftzeichen zur Übertragung vom Computer zum Drucker oder Laserbelichter.

RTF

Rich Text Format. Ein Format für den Dateiaustausch, bei dem Informationen über die Schrift, Schriftgröße, den Schriftstil und die Stilvorlagen (bei Anwendungen, die Stilvorlagen unterstützen) erhalten bleiben.

TTF

True Type Font.

Type-1-Schriften

Schriften, die ihre Beschreibung in Form von PostScript-Befehlen speichern. Das von Adobe entwickelte Format ist zu einem Standard in der Druckvorstufe geworden.

9. Anhang 2: Hinweise zur Nutzung von CSS für veraltete Browser (3., 4. Generation)

- Literatur:
  - Nedregård, Stephan: Web Workshop.
    http://www.micropop.com/workshop/

9.1. Implementation in Bowsern

- Netscape 1.x, 2.x, 3.x: Keine Unterstützung.
- Netscape 4.x: Fehlerhafte Unterstützung.
- Gecko (Mozilla Project), Netscape 6.x: Vollständige und (fast) fehlerfreie Unterstützung.
- Internet Explorer 3.x: Eingeschränkte und stark fehlerhafte Unterstützung.
- Internet Explorer 4.x und später: Relativ fehlerfreie Unterstützung.
- Opera 3.5, 4, 5, 6.

9.2. Wichtige Grundsätze bei der Anwendung

- Testen Sie die Style-Sheet-Angaben mit den gängigen Browsern.
- Ein Minimum an Angaben erspart Ihnen unter Umständen separate Definitionen für die verschiedenen Browser. Verzichten Sie vorläufig auf Randangaben.

9.3. Problem Internet Explorer 3.0

CSS wurden in den Internet Explorer bereits zu einem Zeitpunkt versucht zu implementieren, als dies noch kein Standard war. Das größte Ärgernis bereitet die fehlende Unterstützung für Hintergrundfarben.

Es wird daher allgemein empfohlen, beim Internet Explorer 3.x auf Stilvorlagen zu verzichten. Dies ist relativ einfach: ein ausnutzbarer Fehler beim Internet Explorer 3.x: Er nimmt immer die erste Deklaration, nicht die vorgeschriebene letzte (das ist ja eigentlich der Sinn von Cascading Style Sheets). Das lässt sich ausnutzen, indem man zuerst die Angaben für IE 3.x, dann die Angaben für alle anderen Browser notiert:

```
.style {color: black;}
.style {background: black; color: white;}
```

9.4. Problem Netscape 4.x

9.4.1. Hintergrundfarben

Bei einer Reihe von Tags funktioniert die Angabe von Hintergrundfarben nicht, leider auch bei dem so wichtigen `<div>`-Tag (Tabellen funktionieren). Hintergrundfarben sind so häufig nur bei Text, der eine Hintergrundfarbe zugewiesen bekommen hat, sichtbar, nicht aber im gesamten Element.

Um dies dennoch zu erreichen, muss man die Angabe der Hintergrundfarbe mit `margin`- und `border`-Angaben kombinieren, letztere Angaben können mit beliebigen Werten kombiniert werden:

```
.style {background-color: yellow; margin: 0px; border: none}
```

In jedem Fall werden beide Angaben benötigt. Wird anstelle der `margin`-Angabe `width: 100%` benutzt, erhält man eine ähnliche, aber nicht so genaue Darstellung. Es ist aber in jedem Fall zu prüfen, ob die realen Ränder mit den gewünschten übereinstimmen. Unter Umständen führt dieses Verfahren aber auch dazu, dass die Standard-Fontangaben nicht mehr eingehalten werden.

9.4.2. Fontangaben mit border-Angaben

Wenn einige Tags wie `<h1>` bis `<h6>`, `<p>` oder `<b>` mit `border`-Angaben versehen werden, so werden die Standard-Schriftangaben, die für `<body>` definiert wurden, nicht benutzt.

Geben Sie als Lösung für alle diese Tags Fontangaben an, und setzen Sie den Text konsequent zwischen `<p> ... </p>`-Tags.

9.4.3. Fehlende Font-Auszeichnung bei Veränderung der Fenstergröße

Die Font-Auszeichnungen können möglicherweise verschwinden, wenn die Fenstergröße des Navigators verändert wird.

Hier hilft, wenn JavaScript erlaubt ist, die Ergänzung des `<body>`-Tags:

```
onresize="if (document.layers) window.location.reload();"
```

9.4.4. Padding- und border-Problem

Padding

Wenn Stilanweisungen der folgenden Art für das `<body>`- oder ein anderes Tag vereinbart werden, wird das Padding (Auspolsterung) des entsprechenden Elements fehlerhaft vorgenommen:

```
position: absolute; top: 0px; left: 0px;
border: 10pt gray solid;
border-width: 0pt 0pt 0pt 0.5cm; margin: 0;
padding: 2cm;
```

- Problemlösungen:
  Benutzen Sie kein Padding am unteren Rand, geben Sie `padding-bottom: 0px` an.

Randproblem

Netscape unterstützt keinen linken Rand. Es gibt keine Lösung außer die Verwendung von Tabellen.

9.4.5. Hintergrundbilder

Die Relativ-Pfadangabe im Falle von Hintergrundbildern wird fälschlicherweise auf des HTML-Dokument, nicht aber auf die Formatvorlage bezogen.

```
body { background: url(back.gif) }
```

Als Ausweg bieten sich entweder absolute Pfadangaben an oder die Abspeicherung des Hintergrundbildes an allen notwendigen Stellen.