Cascading Style Sheets (CSS)

CSS sind eine unmittelbare Ergänzung zu HTML. Es handelt sich um einen firmenunabhängigen, offen dokumentierten und frei verwendbaren Standard.
Es ist eine Sprache zur Formatierung einzelner HTML-Tags.
Sie ermöglicht zum Beispiel das Festlegen von Schriftart, -größe und -stil für beliebige Tags, aber auch die Definition ihres Abstandes zum Bsp. zum nächsten Absatz. Genauso ist es möglich einzelne Elemente mit Hintergrundfarben oder -bildern zu hinterlegen, oder auch einzelne Elemente der Seite punktgenau zu positionieren u.v.a.m.
Wer richtig Feuer gefangen hat, kann sich ja mal Quelle 5 ganz, ganz ruhig zu Gemüte führen. Ihr werdet über die Mächtigkeit dieser “HTML-Erweiterung” erstaunt sein. Die Möglichkeiten, die man hier eröffnet bekommt, lassen sich bestenfalls im Ansatz erahnen.

Der Clou ist aber, dass diese CSS zum einen zentral in einer oder in mehreren Dateien verwaltet werden können, zum anderen auch auf einen einzelnen Tag wirken können.
Damit ist es möglich eine zentrale Formatvorlage zu definieren, die dann in alle Seiten einer Homepage eingebunden wird. Somit ist es relativ simpel, das Design der Seite schnell mal zu ändern.

CSS wird derzeit in Level 2 genutzt.
Level 3 befindet sich in der Entwicklung.
Das W3C favorisiert eindeutig CSS zur Formatierung von HTML-Dokumenten.
In Zukunft sollen in den Browsern nur noch CSS akzeptiert werden. D.h. der <font>-Tag wird eindeutig zum Auslaufmodell.

Einschränkung:
Ein Standard ist eine feine Sache, wenn er allgemein bekannt ist und sich alle daran halten. Schade ist es jedoch, wenn ein "Designer" eine ausgefeilte Seite kreiert, jedoch die Browser dann das Design verhunzen. Zwei Gründe gibt es hier: Erstens nutzt der geneigte Besucher der Webseite einen alten Browser, der CSS nicht oder nur rudimentär unterstützt und zweitens sind die css-Eigenschaften nicht alle gleich, vollständig und richtig in neueren Browsern implementiert.
Um die gewünschten Effekte zu erzielen hat der Web-Designer halt die Aufgabe, seine Seiten in verschiedenen (neueren) Browsern zu testen, was mitunter gar nicht so einfach ist, da man i.d.R. systembedingt gewisse Browser gar nicht testen kann (Konqueror von KDE, Safari von Apple).
Folgerung: Eine gewisse Zurückhaltung bei der Gestaltung und eine umfassende Information vor Beginn der Arbeit sind sicher nicht von Schaden und sparen womöglich sogar unnütze Arbeit.

Quellen:

1.:	Tutorial:	**http://www.webdesign-referenz.de/html_css.shtml**
1.1.:	Die ganze Tutorial-Seite:	**http://www.webdesign-referenz.de/start.shtml**
2.:	Tutorial:	**http://www.schattenbaum.net/css/index.php**
3.:	Tutorial:	**http://www.style-sheets.de/css/index.html**
4.:	Stefan Münz**:**	**SELFHTML Version 8.1**
5.:	CSS von W3C (deutsch):	**http://www.edition-w3c.de**
6.:	CSS-Referenz:	**http://www.mediaevent.de/css**
7.:	Browserkompatibilität:	**http://css4you.de/browsercomp.html**

Drei Möglichkeiten CSS zu definieren

In einer externen Datei, die im Kopf der HTML-Datei gelinkt wird:

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

Im Kopf der HTML-Datei als Style-Abschnitt:

```
<head>
    <style type="text/css">
    <!--
    Style-Definitionen
    -->
    </style>
</head>
```

Als Attribut in einem Tag:

```
<p style="font-family: Verdana; font-size: 10pt;">
   Zu formatierender Text
</p>
```

Syntax einer CSS-Anweisung:

NAME {FORMATIERUNG**1**; FORMATIERUNG**2**;...}
dabei kann NAME entweder ein **TAG** oder ein **KLASSENNAME** sein.

Beispiele für Tags:

```
p {font-family: Verdana; font-size: 10pt; font-color: ivory;}
a {text-decoration: none}
```

Beispiele für Klassen
(kann man sich als allgemeingültige Layoutanweisungen vorstellen, die nicht an einen speziellen Tag gebunden sind):

```
.innen {font-color: ivory;}
.aussen {font-color: blue}
```

```
<a href="seite3.html" class="innen">3. Seite</a>
<a href="seite5.html" class="aussen">5. Seite</a>
```

Erweitertes Farbmodell:

Das normale HTML-Farbmodell gilt auch für CSS, jedoch wurde es um die Möglichkeit erweitert, die Rot-, Grün- und Blaubestandteile der jeweiligen Farbe auch dezimal oder in Prozent anzugeben.

Die Formatierung sieht dann so aus: **rgb (rrr, ggg, bbb)**

Beispiel:

```
<span style="color: rgb(255,255,0);">gelber Text</span>
```

Schriftformatierung (Auswahl):

Schriftart:	font-family:	Schriftarten
Schriftgröße:	font-size:	**SMALL, MEDIUM, LARGE** oder in Punkt: **10PT**
Schriftfarbe:	color:	siehe Farbmodell
Schriftstil:	font-style:	**ITALIC, NORMAL**
Schriftgewicht:	font-weight	**BOLD, BOLDER, LIGHT**
Wortabstand:	word-spacing:	Angaben jeweils in **PX** (wie Pixel)
Zeilenabstand:	line-height:	
Zeichenabstand:	letter-spacing:	
Text-Dekoration:	text-decoration	**UNDERLINE, OVERLINE, LINE-THROUGH, BLINK, NONE** (nicht alle klappen in allen Browsern)
Textumformungen:	text-transform:	**CAPITALIZE, UPPERCASE, LOWERCASE, NONE**
Textausrichtung:	text-align:	**LEFT, CENTER, RIGHT, JUSTIFY**
Textausrichtung:	vertical-align:	**SUB, SUPER, BASELINE, TOP, BOTTOM, MIDDLE, TEXT-TOP, TEXT-BOTTOM**

Um einzelne Stellen in einem Text zu formatieren, nimmt man den -Tag.
Dieser definiert im Prinzip nur einen zusammenhängenden Bereich und bietet somit die Möglichkeit, ohne Zeilenumbruch etc. eine Textstelle umzufärben.
Beispiel:

```
<p>Das ist ein Text mit zwei <span style="color: yellow;">gelben
Wörtern</span> in der Mitte.</p>
```

Hintergrundeinstellungen

Hintergrundfarbe:	background-color:	Farbe
Hintergrund fixieren:	background-attachment:	**FIXED, SCROLL**
Hintergrundbild:	background-image:	**URL( )**
Hintergrundbild positionieren:	background-position:	**LEFT, RIGHT, CENTER, TOP, BOTTOM, CENTER**
Hintergrund wiederholen:	background-repeat:	**NO-REPEAT, REPEAT, REPEAT-X, REPEAT-Y**

Größe von Elementen

Breite:	width:	Angaben jeweils in **PX**
Höhe:	height:	
Überlauf:	overflow:	**VISIBLE, HIDDEN, SCROLL, AUTO**

Abstände zu benachbarten Elementen

oberer:	margin-top:	Angaben jeweils in **PX**
unterer:	margin-bottom:	
linker:	margin-left:	
rechter:	margin-right:	

Beispiel:

```
<style type="text/css">
<!--
    body{margin-top:30; margin-bottom: 0; margin-left: 50; margin-right: 0;}
-->
</style>
```

Abstände zum eigenen Rahmen

oberer Abstand:	padding-top:	Angaben jeweils in **PX**
unterer Abstand:	padding-bottom:	
linker Abstand:	padding-left:	
rechter Abstand:	padding-right:	

Rahmenformatierungen

Rahmenfarbe:	border-color:	Farbe
Linienstiel:	border-style:	**NONE, DOTTED, DASHED, SOLID, DOUBLE, GROOVE, RIDGE, INSET, OUTSET**
Linienstärke:	border-width:	**THIN, MEDIUM, THICK,** Angabe in **PX**

Jede dieser Formatierungen kann auf die einzelnen Seiten eines Rahmens (top, left, right, bottom) angewendet werden. Damit ist es möglich, z.B. Linien innerhalb von Tabellen individuell zu gestalten.
Beispiel:

```
<td style ="border-top-color: #ffcc33; border-bottom-style: dotted"> ... </td>
```

Listenformatierungen

Einrückung:	list-style-position:	**INSIDE, OUTSIDE** (outside erzeugt die normale Liste, in der die Inhalte der Listenelemente linksbündig untereinander stehen. Die Aufzählungszeichen schauen heraus.)
Aufzählungsgrafik:	list-style-image:	**NONE, URL()**
Aufzählungszeichen	list-style-type:	**NONE, CIRCLE, SQUARE, DISC, DECIMAL, LOWER-ROMAN, UPPER-ROMAN, LOWER-ALPHA, UPPER-ALPHA** (klappen in jedem Browser) **LOWER-GREEK, DECIMAL-LEADING-ZERO, HEBREW, ARMENIAN, GEORGIAN, CJK-IDEOGRAPHIC, HIRAGANA, KATAKANA, HIRAGANA-IROHA, KATAKANA-IROHA** (nicht alle klappen im MS-IE)

Das Anfangselement von "ol"-Listen kann mit dem Startattribut festgelegt werden.

Beispiel:

```
<ol start="5" style ="list-style-type: disc; list-style-position: outside">
   <li> ... </li>
</ol>
```

Tabellenformatierungen

Tabellenlayout:	table-layout:	**AUTO, FIXED**
Tabellenrahmen:	border-collapse:	**COLLAPSE, SEPARATE**
Tabellenrahmenabstände:	border-spacing:	Angabe in **PX**
Tabellenüberschrift:	caption-side:	**TOP, BOTTOM**

Leider klappen wieder nicht alle Formatierungen im MS-IE. Also sollte man derzeit wohl besser auf die HTML-Attribute des "table"-tags zurückgreifen.

Pixel-genaues Positionieren von Elementen

Positionierung:	position:	ABSOLUTE, FIXED, RELATIVE, STATIC
zu den Rändern:	top:	Angaben jeweils in PX
	bottom:	
	left:	
	right:	

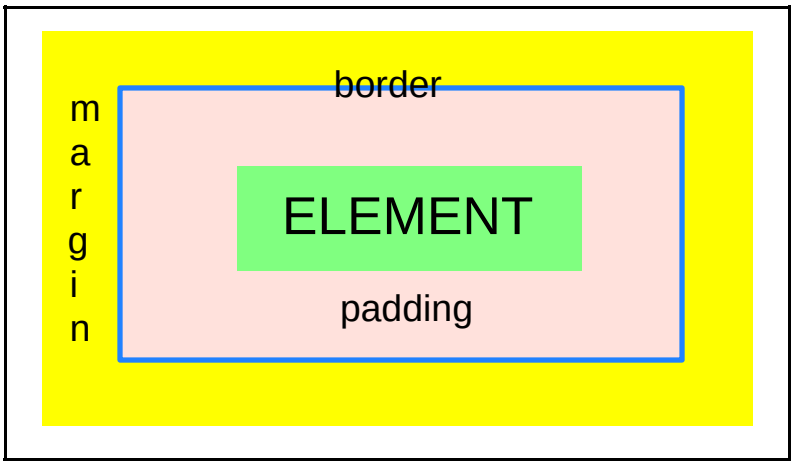
Zur Definition des zu positionierenden Bereiches wird der - tag verwendet.

Beispiel:

```
<div style ="position: absolute; left: 20px; top: 300px;"> ... </div>
```

Boxmodell

CSS erzeugt für ein Elemement eine Box.
Das Element selbst kann mittels `width` und `height` in seiner Größe formatiert werden.
`padding` ist der Abstand zwischen dem Element und dem Rand (`border`)
`margin` gibt den Abstand zur nächsten Box an



Vererbung

Tags, die von anderen eingeschlossen werden, erben deren Eigenschaften (Eltern und Kindelemente: Man kennt das ja: "Solange Du Deine Beine unter unseren Tisch ...")

Containermodell

Mit Hilfe des -Tags werden sog. Container definiert, die quasi frei auf einer Seite positioniert werden können. (Dadurch werden Tabellen in Zukunft ihre Bedeutung als Designelement verlieren und wirklich nur noch zur Darstellung tabellarischer Inhalte genutzt werden.)
Mit Hilfe der Eigenschaft z-index können mehrere Container übereinander gestapelt werden, was natürlich dem Designer verglichen mit Tabellen ganz andere Möglichkeiten eröffnet. Jedoch ist auch hier die Unterstützung durch den MS-IE nicht so, wie sie sein sollte.