

1. Allgemeine Infos zu CSS

CSS sind eine Erweiterung zu (X)HTML und sind zuständig für das Layout und die Formatierung einzelner (X)HTML-Elemente. Es ist möglich, moderner Browser vorausgesetzt, das ganze Layout inkl. der Positionierung mit CSS zu gestalten.

2. Was ist CSS?

CSS ist eine direkte Erweiterung zu HTML und hat die Aufgabe die einzelnen HTML-Elemente (Markierungen) zu formatieren. Also, die Größe, Farbe, Positionierung etc. eines (X)HTML-Elementes wird bzw. sollte durch CSS geregelt werden – vorausgesetzt man will seine Seiten in einem guten und (mittlerweile) auch gültigen (X)HTML-Code schreiben. So gesehen hilft CSS (X)HTML dabei die ursprüngliche (und die eigentliche) strukturierende Aufgabe zurückzuerlangen.

3. Warum sollte man CSS nutzen?

Wie schon oben erwähnt, sollte man CSS für die Formatierung und das Layout nutzen, denn dafür wurde es 1996 von W3C verabschiedet. Aber man nutzt die CSS nicht deswegen, weil W3C sie "erfunden" hat, sondern aus vielschichtigen und sehr wichtigen Gründen.

1. Der erste Grund ist, daß die CSS, schon alleine in der Version 1.0 von 1996, mehr Möglichkeiten zur Formatierung bietet als dies bei HTML der Fall ist. Als Beispiel könnte man die versch. Effekte nennen, die entstehen wenn man mit dem Cursor über einen Link geht (der sog. "hover"-Effekt) oder verschiedene Rahmentypen, Positionierung von Elementen etc.
2. Der zweite Grund ist die Einsparung an Quellcode, was vor allem den großen Seiten zugute kommt. Denn oft vorkommende Formatierungen können in einer externen CSS-Datei ausgelagert werden und tauchen somit nicht mehr im Hauptdokument auf.
3. Der dritte Grund erschließt sich aus dem zweiten Grund. Es handelt sich dabei um die Übersichtlichkeit des Quellcodes. Denn wenn die ganzen Formatierungen nicht mehr in der Hauptdatei auftauchen, wird der Code automatisch übersichtlicher.
4. Der vierte Grund ist die totale Trennung vom Inhalt (und seiner Struktur) und dem Layout (und den Formatierungen). Das spart nicht nur den Quellcode sondern macht eventuelle Änderungen am Layout einfacher, da alle Formatierungen zentral verwaltet werden.

Fazit: wenn du CSS nutzt sparst du Arbeit und Zeit, die Arbeit ist übersichtlicher, du hast mehr Möglichkeiten und gleichzeitig hältst du dich an die von W3C verabschiedeten Standards.

4. Welche CSS-Versionen gibt es?

Es gibt momentan zwei Versionen von CSS. Die Version CSS 1.0 von 1996 und die Version CSS 2.0 von 1998. Zur Zeit wird an der Version CSS 3.0 gearbeitet. Die Version 2.0 wird momentan von W3C empfohlen und baut auf der Version 1.0 auf.

5. CSS lernen

CSS ist relativ leicht zu lernen, auch wenn dir das am Anfang nicht so erscheint. Das ist u. a. deswegen, weil es 'nur' eine Ergänzung zu (X)HTML ist und zwar eine Ergänzung, die für die Formatierung und das Layout zuständig ist. Aber auch CSS kann anspruchsvoll sein, vor allem wenn es um pure und tabellenlose CSS-Layouts geht. Die Schwierigkeit liegt aber weniger an CSS als vielmehr an den mangelnden und fehlerhaften Interpretationen der Browser. Auch an kostenlosem Werkzeug mangelt es nicht, Phase 5, PsPad und NVU sind gute HTML-Editoren und haben auch eine CSS-Syntaxhervorhebung. Wenn man es spartanisch mag, dann kann man auch auf das Notepad zurückgreifen.

6. CSS-Datei erstellen und einbinden

Eine CSS-Datei erstellt man relativ einfach. Entweder man nimmt das Notepad, Phase5 oder PsPad. Wichtig ist, dass es ein Texteditor ist, der keine Formatierungen macht, also kein Word o. ä.

Dann schreibt man die Formatierungen und speichert diese Datei als "dateiname.css" ab.

Bei Notepad: /Datei /speichern unter /alle Dateien/beispiel.css.

Bei PsPAD geht es noch einfacher: /Datei /speichern unter/dateiname.css

Die CSS-Dateien werden im Kopfbereich der HTML-Dateien eingebunden und zwar mit:

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

Es gibt mehrere Möglichkeiten wo eine CSS-Anweisung angebracht werden kann:

- als eine sog. 'inline'-Anweisung (z. B. **<p style="font-weight:bold;">**),
- als ein interner globaler Stil im Kopfbereich der HTML-Datei oder
- als eine externe Datei (in meisten Fällen die sinnvollste Lösung).

Allerdings empfehle ich den Einsatz von einer externen CSS-Datei, was vor allem bei größeren Webprojekten sinnvoll ist. Denn CSS hat u. a. die Aufgabe die Formatierungen zu zentralisieren und zu vereinheitlichen. Die inline und internen Anweisungen sollten nur verwendet werden, wenn man an einer bestimmten Stelle die Formatierung anders haben will. Denn die Anweisungen im Dokument haben eine höhere Wertigkeit und überschreiben daher die Anweisungen eines externen Dokuments. Es gilt: inline überschreibt intern (global), intern überschreibt extern. Die externe CSS-Datei kann man wiederum auf mehreren Wegen einbauen, von denen ich allerdings nur auf zwei eingehen werde, da sie am häufigsten vorkommen. Es handelt sich hierbei entweder um die verlinkte oder die importierte Lösung. Mittels

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

wird eine externe CSS-Datei verlinkt. Diese Anweisung wird im Kopfbereich (zwischen <head> und </head>) untergebracht. Dabei sollte auf den richtigen Pfad zu der Datei geachtet werden. Diese Anweisung verstehen auch alte Browser wie z.B. Netscape 4.x und Internet Explorer 4.x (Mac und Win).

7. Aufbau der Stylesheet-Angabe

Eine Stylesheet-Angabe besteht aus der Eigenschaft und dem Wert, getrennt durch einen Doppelpunkt. Die Eigenschaft ist das, was du formatieren möchtest. Z.B.: *color* für die Schriftfarbe, *border-width* für die Rahmenbreite oder *text-decoration* für Unterstrichungen. Als Wert sind Farbangaben, Einheiten oder Schlüsselworte möglich. Hier einige Beispiele:

- **Schriftfarbe rot:** color:#ff0000;
color ist die Eigenschaft für die Schriftfarbe und *#ff0000* ist der Wert als hexadezimale Farbangabe für Rot
- **Schriftgewicht fett:** font-weight:bold;
font-weight ist die Eigenschaft für das Schriftgewicht und *bold* ist der Wert als Schlüsselwort für fett
- **Unterstrichen:** text-decoration:underline;
text-decoration ist die Eigenschaft für Unter- und Überstrichungen und *underline* ist der Wert als Schlüsselwort für unterstrichen
- **Elementhöhe:** height:100px;
height ist die Eigenschaft für die Höhe und *100px* ist der Wert als Zahl mit der Einheit Pixel. Wichtig! Zwischen der Zahl und der Einheit darf kein Leerzeichen stehen.

8. Beispiel einer CSS-Datei

Dreispaltiges Layout mit Kopf- und Fußzeile.

```
/* Fuer das ganze Dokument geltende Formatierungen */
body
{
```

```

background-color: #000000;
color: #ffffff;
font-family: verdana, sans-serif;
font-size: 12px;
}
h1
{
font-size: 18px;
border: 1px solid #00FF00;
text-align: center;
background-color: #ffffff;
color: #000000;
padding: 1px;
}
.fett
{
font-weight: bold;
}
/*dreispaltiges Layout mit Kopf- und Fusszeile*/
#rahmen { margin: 0px auto;
width: 800px;
height: 900px;
}

#header { border-bottom: 1px solid rgb(0, 0, 0);
margin-bottom: 5px;
background-color: rgb(255, 255, 255);
height: 100px;
}

#links { float: left;
width: 14%;
background-color: rgb(255, 235, 205);
height: 100%;
}

#content { float: left;
width: 66%;
margin-left: 3%;
background-color: rgb(220, 220, 220);
min-height: 100% ! important;
height: 100%;
}

#rechts { float: right;
width: 14%;
background-color: rgb(204, 204, 204);
height: 100%;
min-height: 100%;
}

#fusszeile { border-top: 1px solid Black;
clear: both;
padding-top: 10px;
text-align: center;
}

h1.header {
padding-top: 30px;
}

```

Ein zentrierter <div>-Container für alles

Am Einfachsten ist es einen <div>-Container auf der Seite zu zentrieren und in diesem den gesamten Seiteninhalt einzufügen. Und das geht so:

```
#rahmen {
margin: 0px auto;
width: 800px;
height: 900px;
}
```

width:800px legt die Breite auf 760 Pixel fest und

margin:0px auto bestimmt den Außenabstand. Gibst du für *margin* wie in diesem Fall zwei Werte an, so legt der erste jeweils den oberen und unteren Abstand und der zweite den linken und rechten Abstand fest. Der Wert *auto* besagt, dass der Browser den Abstand automatisch berechnen soll und zwar für links und rechts gleich, da ja nur ein Wert vorgegeben ist. Und gleiche Außenabstände haben nur zentrierte Objekte, oder?

Die Eigenschaft [float](#) plaziert Elemente auf der Seite und lässt andere vorbeifließen. In diesem Fall wird dem Logo `float:right` zugewiesen. Das bedeutet, dass das Logo rechtsbündig plaziert wird und der folgende Inhalt, in diesem Fall das <h1>-Tag, links vorbeifließt, bzw. links steht.

Mit [clear](#) beendest du diesen Effekt, d.h. für das erste Element, das nicht mehr floaten soll, musst du `clear` notieren. Deshalb ist in den Stylesheet-Angaben für den <div>-Container des Menüs u.a. folgendes eingefügt:

```
#menu {
clear:left;
border-top:1px solid #669999;
}
```

Jetzt speicherst du diese Datei als "irgend_ein_name.css" ab.

Was haben wir jetzt mit dieser Datei erreicht?

Der Body-Bereich bezieht sich auf das ganze Dokument. Jedes (X)HTML-Dokument, dem du dieses CSS zuweist bekommt einen schwarzen Hintergrund, eine weiße Schrift, die entweder Verdana oder einem seriflosem Schrifttyp angehört und die 12 Pixel groß ist.

Darüber hinaus hat jede Überschrift erster Ordnung (h1) in diesen Dokumenten eine Schriftgröße von 18 Pixeln, eine feine grüne Umrahmung, zentrierte Schrift, weiße Hintergrundfarbe, schwarze Schriftfarbe und die Schrift hat einen zusätzlichen Abstand zum Rahmen von einem Pixel.

Mit **.fett** erreichst du eine weitere Eigenschaft von CSS, die Klassenbildung. Denn alle HTML-Elemente, die du so klassifizierst (z. B. `<p class="fett">Text</p>`) bekommen eine fette Schrift.

Mit **#header**, **#links**, **#rechts**, **#content** und **#Fußzeile** bestimmst du einmalige Bereiche. Du kannst so eine ID nur einmal vergeben. Der Vorteil von einer ID gegenüber der Klasse ist, dass man die ID mittels eines Links erreichen kann.

9. CSS-Rahmen

Die Angaben zum Rahmen eines HTML-Elementes können nach folgendem Muster entweder in eine externe CSS-Datei ausgelagert werden, im Kopf der HTML-Datei oder im Element selber getätigt werden.

```
element{
    rahmenbreite rahmenart rahmenfarbe
}
```

oder als inline-Element:

```
<element style="rahmenbreite rahmenart rahmenfarbe;">Inhalt</element>
```

Rahmenangaben

- `border-width` – Angabe zu Rahmenbreite (Absolute oder relative Größenangaben)
- `border-style` – Angabe zu Rahmenart (s. u.)
- `border-color` – Angabe zu Rahmenfarbe (Namen, RGB- oder Hexadezimal-Werte.)

Verschiedene Rahmentypen:

- `solid` – Durchgezogene Linie
- `dotted` – Gepunktete Linie
- `dashed` – Gestrichelte Linie
- `double` – Doppelte Linie
- `groove` – 3-D, Linie mit einem "Graben" in der Mitte.
- `ridge` – 3-D, Anhebung in der Mitte.
- `inset` – 3-D, "eingedrückter Knopf"
- `outset` – 3-D, "aktiver Knopf"
- `none` – Kein Rahmen

Damit dies alles etwas verdeutlicht wird, gibt es einige Beispiele. Nehmen wir mal an wir haben einen Textabsatz und dieser soll einen schwarzen Rahmen bekommen. Dieser schwarze Rahmen soll sehr dünn sein und es soll eine durchgezogene Linie sein.

Hier das erste Beispiel mit einem inline-CSS:

So sollte die Seite dann etwa aussehen:



Aufgabe:

1. Lege einen Ordner „css-test“ an. In diesem Ordner sollen zwei weitere Ordner „css“ und „images“ angelegt werden.
2. Erzeuge eine neue css-Datei in PsPad und speichere sie im Ordner „css“ als „csstest“.
3. Schreibe nun das gesamte Beispiel-Stylesheet ab und speichere es.
4. Erzeuge eine neue Html-Seite in PsPad.
5. Speichere sie im Ordner „css-test“.
6. Übertrage den Html-Quelltext und speichere die Datei.
7. Fülle auf der neuen Seite die „DIV-Container“ mit beliebigem Inhalt und sieh die Seite im Standardbrowser an.
8. Beseitige die Fehler.
9. Gut gemacht!