CSS

Grundlagen

CSS-Dokumente in HTML einbinden

CSS kann auf 3 Arten eingebunden werden:
- in einer separaten CSS-Datei
- im Kopfbereich des HTML-Dokuments
- im einzelnen HTML-Element

Separate CSS-Dateien

Mit einem Standalone-Tag im head Bereich kan eine beliebige CSS-Datei referenziert werden. Besonders nützlich wenn gleiche CSS-Eigenschaften für mehrere HTML-Dokumente gelten sollen.

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

CSS im HTML-Head

Definieren eines Bereichs im head für CSS-Formate.

```
<head>
        <style type="text/css">
                Zwischen den style-Tags können
                CSS-Befehle definiert werden die
                für das HTML-Dokument gelten
        </style>
</head>
```

CSS in HTML-Elementen

Mit dem globalen Attribut style können CSS-Befehle direkt im HTML-Tag definiert werden.

```
<p style="font-size: 2em; background-color:
#808040;">....</p>
```

Die CSS-Befehle werden mit einem Semikolon abgeschlossen!

TIPP: Die verschiedenen Varianten um CSS in eine HTML-Dokument einzubinden können miteinander kombiniert werden. Eine Style-Definition in einem HTML-Tag hat hierbei vorrang vor einer in einer separaten Datei definierten CSS-Datei!

Syntax und Grammatik von CSS

CSS Befehle werden nach folgendem Schema notiert:

```
Selektor { Die Eigenschaften werden in
           Blöcken von Klammern umgeben
           Eigenschaft-1: Wert;
           Die Eigenschaften enden mit einem
           Semikolon!
           Eigenschaft-2: Wert;
           usw.}
```

Die Wertangaben unterscheiden sich je nach Art der Eigenschaft: Prozent, Pixelangaben, hexadezimale Farbangaben etc.

CSS-Regeln

In CSS sind Regeln an einem führenden @-Zeichen zu erkennen. Sie legen die Rahmenbedingungen für die CSS-Befehle im Dokument fest. Zwar sind sie nicht zwingend erforderlich, jedoch sehr nützlich wie z.B. @ media.

```
@import url(beispiel.css);
Importieren einer separaten CSS-Datei
@charset UTF-8;
Erzwingen eines bestimmten Zeichensatzes
für alle nachfolgenden Befehle
@page {...}
Seitenformate zur Ausgabe auf Druckern
@media screen {font-family: Arial;}
Legt für bestimmte Ausgabeformate CSS-
Befehle fest
@media screen and (max-width: 300px) {...}
Nur bei bestimmter Ausgabebreite
```

Weitere Beispiele für Ausgabetypen für @media:
all (alle Gerätetypen), handheld (Smartphones), print (Drucker), speech (Sprachsynthesizer), tv (Fernseher)

Beispiele für die erweiterte Medienabfrage:

```
@media (orientation: portrait){}
Angaben für hochformatige Ausgabe (Smart-
phones, Drucker)
@media not screen and (color){}
Angaben nicht für Bildschirme, sondern für
Farbdrucker
```

Selektoren

Bei Style-Angaben im HTML-Tag nicht nötig, da diese stets für das jeweilige Element gelten. Bei externen CSS-Dokumenten sind Selektoren jedoch unverzichtbar.

```
p {                                 gilt für alle p Elemente des Dokuments
        font-family: arial;         Schriftart Arial
        color: blue;                Schriftfarbe blau
        background-color: yellow;   Hintergrundfarbe gelb
}
```

Um CSS-Eigenschaften für alle Elemente eines bestimmten Typs zu definieren wird einfach der HTML-Elementtyp als Selektor angegeben. Dann gelten die Eigenschaften für alle Elemente dieses Typs.

```
p, li {             gilt für alle p und li Elemente des Dokuments
        color: blue; Schriftfarbe blau
}

* {color: blue;}    Der Universalselektor * legt für alle Texte eine blaue Schrift fest
```

TIPP: Verschachtelte Elemente „erben" automatisch die meisten CSS-Eigenschaften des Eltern-elements. Z.B. die Schriftfarbe.

Verschachtelte Selektoren

```
p a {               Nur a-Elemente die in einem p-Element liegen werden selektiert
        color: green; }
p * a {             Nur a-Elemente in einem weiteren Element und in einem p-Element
        color: green; }
p > a {             Nur a-Elemente ohne ein Element dazwischen in einem p-Element
        color: green; }
p + a {             Nur a-Elemente die direkt auf ein p-Element folgen
        color: green; }
```

Klassen und ID Selektoren

.blau {
Elemente mit der Klasse blau (class="blau") werden selektiert }

#blauerBereich {
Elemente mit der ID blauerBereich werden selektiert }

Pseudo-Elemente und Pseudo-Klassen

```
a: link {...}      Für noch nicht besuchte Links
a: visited {...}   Für bereits besuchte Links
a: hover {...}     Wenn sich der Mauszeiger über
                   dem Element befindet
a: active{...}     Wenn ein Verweis angeklickt
                   wird
```

Ursprung und Priorität

Betreffen verschiedene Formatangaben das glei-che Element gilt folgende absteigende Priorität:

1. Benutzer-Stylesheet-Definitionen mit !important
2. Autoren-Stylesheet-Definitionen mit !important
3. Autoren-Stylesheet-Definitionen
4. Benutzer-Stylesheet-Definitionen
5. Browser-Stylesheet-Definitionen

TIPP: Jede CSS-Eigenschaft kann mit !impor-tant versehen werden, wodurch sie Vorrang erhält: p { font-size: 1em !important; }

Spezifität

Sobald Stylesheets ein bisschen länger werden, gibt es früher oder später für ein HTML-Element mehrere CSS-Regeln, die sich zum Teil widersprechen, bzw. überschreiben. Die Frage ist, wie der Browser in solchen Konfliktfällen entscheidet.
Die Lösung: Der Browser berechnet anhand eines einfachen Punktesystems, welcher Selektor der wichtigste ist.

Hierbei gibt es 4 Kategorien:

A gilt für style-angaben die direkt in einem HTML-Tag notiert sind (höchste Kategorie)

B gilt für Selektionen per id-Attribut

C gilt für Selektionen die Klassen oder Pseudoklassen betreffen

D entspricht der Anzahl an selektierten (evtl. verschachtelte Elemente)

Selektor:	A	B	C	D
p a	0	0	0	2
#blauerBereich	0	1	0	0
.blau a	0	0	1	1
style = „ “	1	0	0	0

CSS-Eigenschaften bisher:

font-family: helvetica, arial;	legt von links nach rechts Schriftarten fest
color: blue;	definiert die Schriftfarbe
font-size: 2em/12px;	Schriftgröße (2em = doppelte Standardgröße)
font-weight: bold/700;	Schriftstärke
font-style: italic/normal/oblique;	Schriftstil
background-color: red/#FF0000;	Hintergrundfarbe

Das CSS Boxmodell

Das Boxmodell definiert den Raum, den ein HTML-Element im Browserfenster einnimmt. Dabei werden HTML-Elemente in zwei verschiedene Kategorien unterteilt. In Block- und Inline-Elemente.

Block-Elemente
Diese nehmen in der Breite so viel Raum ein wie möglich und in der Höhe so viel wie Raum wie erforderlich um das Element ganz darzustellen. Außerdem erzeugen Block-Elemente immer eine neue Zeile im Textfluss.

Durch CSS Eigenschaften kann die Breite von Block-Elementen manipuliert werden:

```
<p style="width: 80%; ">...</p>
```

Reduziert die Breite auf 80% des vorh. Platzes

Inline-Elemente
Diese nehmen sowohl in der Breite, als auch in der Höhe nur so viel Platz ein wie erforderlich um das Element ganz darzustellen. Sie erzeugen auch keinen Zeilenumbruch, werden also nebeneinander dargestellt.

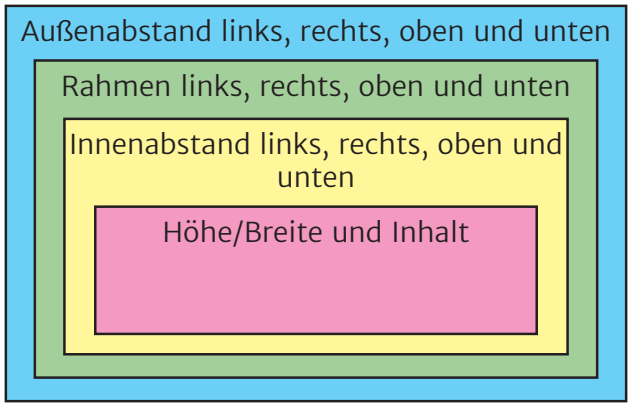
```
<p>Heute ist alles <em style="background: blue;">wirklich</em> super!</p>
```

kein Zeilenumbruch durch das Inline-Element em

TIPP: Mit CSS können Inline-Elemente zu Block-Elementen gemacht werden und umgekehrt: display: block; Das Element wird wie ein Block-Element angezeigt
display: inline; Das Element wird wie ein Inline-Element angezeigt

Elementmaße abschätzen/ kalkulieren

Höhe und Breite eines Elements errechnet sich aus der Addition von:
Breite/Höhe des Elementinhalts
+ Innenabstand (padding)
+ Rahmendicke
+ Außenabstand (margin)



CSS-Eigenschaften zur Manipulation der Abstände

```
.tabellenfeld {
width: 100px;          legt die Breite fest
height: 50px;          legt die Höhe fest
margin: 0;             legt den Außenabstand
                       rundherum fest
border: solid 1px black;
Rahmen (durchgezogen, 1 Pixel dick, schwarz)
padding-top: 5px;
legt den Innenabstand oben fest
padding-left: 10px;
legt den Innenabstand links fest
padding-right: 10px;
legt den Innenabstand rechts fest
padding-bottom: 5px;
legt den Innenabstand unten fest
background-color: grey;
Hintergrundfarbe wird grau
}
```

Die border-Eigenschaft kann durch top, left, right oder bottom ergänzt werden (Bsp.: border-right). Abgesehen vom Wert 0 muss immer eine Maßangabe erfolgen. Bei Fließkommazahlen sollte immer die englische Schreibweise (Punkt statt Komma) verwendet werden, also zum Beispiel 55.5%.

TIPP: Bei margin kann der Wert „auto" für eine Zentrierung sorgen:
margin: 0 auto;
Element wird im übergeordneten Element horizontal zentriert.

Hintergrundfarben/Bilder

Farben oder Bilder die als Hintergrund eines Elements in CSS definiert werden füllen nicht den gesamten Bereich des Boxmodells aus. Sondern nur den Bereich des Inhalts und den des Innenabstands.

Positionierte Elemente

Es ist möglich Elemente aus der normalen in der HTML-Struktur festgelegten Positionierung herauszulösen. Dazu steht dient die CSS-Eigenschaft position die durch verschiedene Werte es möglich macht Elemente aus dem Elementfluss zu lösen und gezielt zu platzieren.

```
position: absolute;
orientiert sich am Rand des Elternelement bzw. Dokumentelement (<html>)
position: relative;
orientiert sich an der Position, die das Element normalerweise hätte
position: fixed;
Positionierung direkt an einer gewünschten Stelle im Browserfenster, das Element scrollt im Gegensatz zu den mit relative oder absolute positionierten Elementen nicht mit!
```

Die Position wird durch zusätzliche CSS-Eigenschaften definiert: top, left, right und bottom

```
#verschoben {
position: relative;
top: 20px;
das Element mit der ID verschoben wird um 20 Pixel hoch gerückt
left: 30px;
von seiner ursprünglichen Position aus um 30px nach links
}
```

Umfließen von Elementen

Es ist möglich Block-Elemente nebeneinander zu platzieren. Dazu wird meist die CSS-Eigenschaft float verwendet. Außerdem bietet sie die Grundlage für Responsive-Webdesign.
Als Werte können left, right oder none vergeben werden.

```
float: left;
folgender Inhalt fließt rechts herum
float: right;
folgender Inhalt fließt links herum
float: none;
es wird explizit kein Umfluss erzeugt
```

Die CSS-Eigenschaft float kann dazu genutzt werden um Text um ein Bild fließen zu lassen oder auch um ein Mehrspaltiges Seitenlayout zu erzeugen.