# Kapitel 9 - Teil 3

## *Bilder und Videos*

Inhalte in diesem Kapitel

Externe und interne Stylesheets

Mit den richtigen Selektoren Elemente gezielt auswählen

Unterschiedliches Gewicht der Selektoren

# HTML und CSS verknüpfen

### Externe Datei

<link rel=“stylesheet“ href=“pfad.css“>

rel = Relationship -> sagt dem Browser das es ein Stylesheet ist

href = Pfad zur Datei

auch mehrere Stylesheets möglich

zuletzt stehende Regel setzt sich durch

### Eingebettete Stylesheets

<style> und </style>

im head Bereich

### Inline-Stile

style-Attribut direkt bei Element

<p style=“color: orange; background-color: black“> blabla </p>

Angaben haben Vorrang vor Angaben aus eingebetteten oder externen Stylesheets

### Fazit

Einfacher, wenn Styling in externen zentralen CSS-Datei steht

Bei inline Styling wird der Code aufgebläht

### Externe Stylesheets mit @import einbinden

<style>

@import „pfad.css“;

//weitere Anweisungen

</style>

Schlecht für Performance (Ladegeschwindigkeit!

Wenn Stylesheets aufgeteilt werden sollen ist sass besser! Arbeitet mit CSS-Präprozessoren. CSS kann in kleine Komponenten aufgeteilt werden, bei Umwandlung ein Stylesheet generiert

### Kommentare in CSS

/\* Kommentar \*/

# Elemente auswählen

### Typselektor element

h3 { color: blue; }

Alle Elemente des Typs h3

Tags ansprechen

### Klassenselektor .klasse

Der Punkt verweist auf eine Klasse

<p class=“custom-background“>blabla</p>

<style>

.custom-background { background-color: yellow; }

</style>

Mehrere Klassen werden mit einem Leerzeichen getrennt

<p class=“custom-background custom-styling“>blabla</p>

Nur bestimmte Elementtypen mit der Klasse

h3.custom-styling { . . . }

### id-Selektor #id

<div id=“container“>

#container { . . . . }

IDs darf es in einer Datei nur einmal geben

IDs braucht man für interne Verlinkungen, Formularelemente, …

Besser über Klassen stylen, da wiederverwendbar

### Selektoren gruppieren

p { color: red; }  
h3 { color: red; }  
/\* wird zu \*/  
p. h3 { color: red; }

Selektoren können mit dem Komma gruppiert werden

### Universeller Selektor \*

\* {margin: 0;}

Von allen Elementen wird der Außenabstand entfernt

# Attributselektoren

<input type=“text“ ….>

input[type=“text“] { . . . }

Attributselektoren berücksichtigen die Attribute von Elementen

* + Eckige Klammern, in denen die Angaben zum Attribut stehen
  + Eckige Klammern müssen direkt an dem Element stehen, auf das sie sich beziehen

### element[attribut]

a[id]

wählt alle a-Elemente aus, die ein id-Attribut haben (egal welchen Wert das Attribut hat

### element[attribut^=wert]

a[href^=“http“] { }

alle a-Elemente, die ein Attribut mit dem Namen href besitzen, dessen Attributwert mit der Zeichenkette http beginnt

### element[attribut$=wert]

a[href$=“pdf“] { }

alle a-Elemente, die ein Attribut mit dem Namen href besitzen, dessen Attributwert mit der Zeichenkette pdf endet

### element[attribut\*=wert]

a[href\*=“yahoo“] { }

Alle a-Elemente, die ein Attribut mit dem Namen href besitzen, dessen Attributwert den Wert yahoo beinhaltet

### element[attribut~=wert]

a[href~=“Externer“] { }

Alle a-Elemente, in deren Wert auch das Wort Externer vorkommt

### element[attribute|=wert]

a[title|=“Abb.“] { }

Alle a-Elemente, die ein Attribut mit dem Namen title besitzen, dessen Attributwert mit der Zeichenkette Abb. gefolgt von einem Bindestrich beginnt

# Kombinatoren

### Nachfahrenkombinatoren

Es werden zwei oder mehr Selektoren durch Leerzeichen getrennt hintereinandergeschrieben

nav a { }  
nav ul { }  
article p { }

oder Kombinationen

nav a.aktuell  
.feld input[type=“text“]

damit können in Verschachtelungen verschiedene Dinge angesprochen werden

Kombinatoren beziehen sich auf die Dokumentstruktur

### Verwandtschaft in HTML: Eltern, Nachkommen, Kinder und Geschwister

Auf Abbildung 9.8. verweisen S. 212

Wichtigste Grundprinzip ist Verschachtelung von Elementen

5 wichtige Begriffe:

* + *Wurzelelement*
    - root = html ist das Wurzelelement, oberstes Element
  + *Elternelement*
    - parent = html ist das Eleternelement für head und body, d.h. head und body befinden sich direkt innerhalb von html
  + *Kindelement*
    - child = head und body sind Kindelemente von html  
      body-Element ist das Elternelement für h1, nav, usw.
  + *Geschwisterelement*
    - sibling = h1, nav und article sind Kindelemente von body
    - sind Geschwisterelemente, da sie auf derselben Ebene sind
  + *Nachfahren*
    - descendant = h2, ul, … sind Kindelemente von article, sind aber auch Nachfahren von body und html  
      sind auf einer tieferen Ebene

### Kindkombinator selector > selector

wählt direkte Kindelemente

div > strong  
wählt alle strong-Elemente aus, die Kindelemente eines div-Elements sind

<p>Ein <strong>Absatz</strong></p>  
<div>Noch ein **<strong>Absatz</strong>**</ div >  
<div><p>Und noch ein <strong>Absatz</strong></p></div>

### Nachbarkombinator selektor + selektor

wählt ein Element aus, das direkt auf ein anderes folgt

beide müssen dasselbe Elternelement besitzen

div + p  
wird das Element p gewählt, das direkt auf div folgt

<div><p>Ein Absatz</p></div>  
**<p>Noch ein Absatz</p>**  
<p>Und noch ein Absatz</p>

### Allgemeiner Geschwisterkombinator selektor ~ selektor

wählt ein Element aus, das auf ein anderes folgt, aber nicht unbedingt direkt

div ~ p  
werden alle p gewählt, die auf div folgen

<div><p>Ein Absatz</p></div>  
**<p>Noch ein Absatz</p>  
<p>Und noch ein Absatz</p>**

# Pseudoklassen und Pseudoelemente

### Linkzustände

a:hover  
Maus fährt über Link  
auch mit anderen Selektoren möglich

a:link  
nicht besuchte Links  
Standard: blau

a:visited  
besuchte Links  
Standard: lila

a:focus  
Link formatiert wenn dieser den Fokus über die Tastatur erhält  
Sinnvoll: zB bei Formularelementen

a:active  
Links die gerade geklickt werden  
Maus klicken u halten um Ergebnis zu sehen

Reihenfolge der Linkzustände ist wichtig im CSS

### :focus-within

Wenn Element oder eines seiner Nachfahren geklickt wird

Formular hervorheben, wenn die Benutzer in ein Formularfeld klicken

Zb: wenn input-Element geklickt wird, soll fieldset anders eingefärbt werden

<form>  
 <fieldset>  
 <label for=“name“>Name</label>  
 <input type=“text“ name=“name“ id=“name“>  
 </fieldset>  
</form>  
fieldset:focus-within {  
 background-color: lightgreen;  
}

### :target

Praktisch bei internen Verlinkungen: kennzeichnet Elemente, die durch interne Links angesprungen werden

:target { background-color: pink; }

### Strukturelle Pseudoklassen

:nth-child(2n) {}

* + Jedes Zweite Element
  + Beispiel Tabellen tr:nth-child(2n) !

:nth-child(even) / :nth-child(odd)

* + Gerade bzw. ungerade Elemente werden angesprochen

:nth-child(1)

* + Wählt erstes Element aus
  + Alternative: :first-child

:nth-child(2n+1)

* + Formatiert alle ungeraden Kindelemente
  + Alternative: :nth-child(odd)

:nth-child(4n)

* + Wählt jedes 4. Element aus

:nth-child(2)

* + Wählt genau ein Element aus (das zweite Kindelement)

:nth-child(-1)

* + Wählt das letzte Element aus
  + Alternative: :last-child

Das n steht für eine Zahl, die hochgezählt wird

Es werden immer die jeweiligen Kindelemente ausgewählt

Neben :nth-child existiert auch :nth-of-type()

* + Dabei zusätzlich der Typ des Elements berücksichtigt

Vollständige Liste der möglichen Selektoren: <https://developer.mozilla.org/de/docs/web/css/pseudo-classes>

### Pseudoelemente

:hover, :nth-child():

* + Sog. Pseudoklassen

Pseudoelemente:

* + Können Teile von Elementen formatiert werden

Unterschied liegt in Schreibweise:

* + Pseudoklassen mit einem : (Doppelpunkt)
  + Pseudoelemente mit zwei :: (Doppelpunkte)

::before und ::after

* + Mit CSS Inhalte einfügen
  + Mit ::before am Anfang oder mit ::after am Ende des angegebenen Elements ergänzt
  + Inhalt, der eingefügt werden soll: content: ;
  + Häufig genutzt zur Ergänzung von Icons oder Tricksereien

::first-line

* + Erste Zeile eines Absatzes zum Beispiel

::first-letter

* + Erster Buchstabe des gewählten Elements

::selection

* + Wählt den Text aus, den der Benutzer markiert hat

::placeholder

* + Platzhaltertext in Formularfeldern

### Spezifität verstehen

id-Selektoren haben eine höhere Spezifität als Klassenselektoren (schwer zu händeln bei großen Projekten)

Klassenselektoren haben eine höhere Spezifität als Typselektoren

Ein Selektor mit zwei Elementen hat eine höhere Priorität als ein Selektor mit einem Element

Guter Artikel dazu: <https://blog.kulturbanause.de/2013/06/css-spezifitat/>