

Web Engineering + Design 1

CASCADING STYLE SHEETS (CSS)

Michael Gfeller

Intro



Die Teilnehmer...

- ... kennen unterschiedliche Varianten wie CSS definiert werden kann und können die beste Variante auswählen
- ... können CSS Selektoren interpretieren und anwenden
- ... können geeignete CSS Selektoren für ein Problem definieren
- ... können die Specificity von einer CSS-Regel berechnen
- ... können erkennen, welche CSS Regeln auf bestimmte Elemente angewendet werden
- ... können mit CSS eine Webseite Stylen
- ... kennen die Eigenschaften von CSS und können diese anwenden und erklären
- ... können mit den Chrome Dev. Tools umgehen (Teil CSS)

Die Teilnehmer...

- ... kennen den Unterschied zwischen Pixel und rem/em
- ... kennen das Problem von relativen Einheiten und Eigenschaften-Vererbung.
- ... wissen, welche Eigenschaften vererbt werden.
- ... können unterschiedliches CSS für Druck und Bildschirm definieren.
- ... verstehen was der «Fluss» ist und wie Elemente sich darin verhalten
- ... können Elemente aus dem «Fluss» nehmen und können mit «Position» umgehen
- ... können den Unterschied zwischen den unterschiedlichen «Display» erklären und an Beispielen anwenden
- ... kennen verschiedene Varianten wie HTML Elemente «verstecken» werden können und sind in der Lage diese korrekt anzuwenden.

Die Teilnehmer...

- ... wissen was das «Box Model» ist
- ... können «Padding» und «Margin» korrekt anwenden
- ... können «box-sizing» erklären und korrekt anwenden
- ... kennen den Unterschied zwischen «Border» und «Outline»
- ... können CSS Eigenschaften selbständig nachschlagen und anwenden

■ CSS Einführung

- Warum CSS?
- Was ist CSS?
- CSS definieren
- CSS debuggen

■ Selektoren

- Basics
- Kombinatoren
- Universal Selektor
- Klassen Selektor
- Attribut Selektor
- Pseudo Elemente
- Pseudo Klassen
- Hintergrund Infos

■ CSS Advanced

- Box-Model
- Border / Outline
- CSS-Eigenschaften
- Display
- Position
- Float
- CSS Kaskade
- Einheiten
- Font & Color

■ Beispiele

■ Quellen

■ Ausblick WED2

WARUM CSS?

- **HTML wurde entworfen um Information zu strukturieren**
- **Mit weiterer Popularität wurden die HTML Tags zweckentfremdet bzw. mit Style Tags ergänzt:**
 - font & blink Tag
 - u & i Tag
 - table Tag wird fürs Layouten «missbraucht»
- **CSS wurde entworfen um ein einheitliches und wiederverwendbares Styling Tool zu Verfügung zu stellen.**
 - Mit HTML5 wurden alle Style Tags als deprecated eingestuft und dürfen nicht mehr verwendet werden.
 - z.B. Font Tag: <https://developer.mozilla.org/de/docs/Web/HTML/Element/font>



Veraltet

Dieses Feature ist veraltet. Obwohl es in manchen Browsern immer noch funktioniert, wird von seiner Benutzung abgeraten, da es jederzeit entfernt werden kann. Es sollte daher nicht mehr verwendet werden.

Weshalb CSS: Beispiel ohne CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body bgcolor="aqua">
  <h1><font color="red">Hello <big>world</big></font></h1>
  <p>Mögliche Variante um eine Webseite zu stylen:</p>
  <ul>
    <li><strike>&lt;Font&gt; Tag</strike></li>
    <li><u>CSS</u></li>
  </ul>
</body>
</html>
```

Hello world

Mögliche Variante um eine Webseite zu stylen:

- ~~Tag~~
- CSS

■ Webstorm erkennt deprecated Tags und markiert diese:

```
<h1><font color="red">Hello <big>world</big></font></h1>
<p>Mögliche Variante um eine Webseite zu stylen:</p>
<ul>
  <li><strike>&lt;Font&gt; Tag</strike></li>
  <li><u>CSS</u></li>
</ul>
```

- **Klare Trennung zwischen Struktur und Style-Informationen**
- **Ermöglicht auf unterschiedliche Ausgabegrößen zu reagieren**
 - Fluides Design => WED2
- **Es ist möglich, unterschiedliche Styles für den jeweiligen Output Type zu definieren**
 - Print
 - Screen
- **Styles können auf Gruppen von Elementen angewendet werden**
 - Kein Copy & Paste
- **Styles können in Files ausgelagert werden**

Weshalb CSS: Beispiel mit CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    body{ background: aqua;}

    h1{ color: red; }

    strong{ font-size: 1.2em;}

    .deprecated{ text-decoration: line-through;}

    .important{ text-decoration: underline;}
  </style>
</head>
<body>
  <h1>Hello <strong>world</strong></h1>
  <p>Mögliche Variante um eine Webseite zu stylen:</p>
  <ul>
    <li class="deprecated">&lt;Font&gt; Tag</li>
    <li class="important">CSS</li>
  </ul>
</body>
</html>
```

Hello world

Mögliche Variante um eine Webseite zu stylen:

- ~~Tag~~
- CSS

HTML ohne CSS?

■ Erste Web-Seite kam ohne Styling aus: <http://info.cern.ch/hypertext/WWW/TheProject.html>

■ HTML alleine sieht «unschön» aus

■ z.B. 20min ohne Styles <http://www.20min.ch/>

- [de](#)
- [fr](#)
- [it](#)

[Inhalt A-Z](#)

Suchen



- [Schweiz](#)
 - [Zürich](#)
 - [Bern](#)
 - [Basel](#)
 - [Zentralschweiz](#)
 - [Ostschweiz](#)
- [Ausland](#)
 - [Panorama](#)
- [Wirtschaft](#)

WAS IST CSS?

“Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents.”

Quelle: <https://www.w3.org/Style/CSS/>

■ Cascading Style Sheets

- Cascading
 - Später in der Vorlesung
- Style Sheet - Language
 - Formale Sprachen um das Erscheinungsbild von Dokumenten bzw. Benutzeroberflächen festzulegen.

■ CSS ist ein «living standard»

- Wir immer weiter entwickelt
- Aktuell ist CSS3
 - „*There will never be a CSS4*“: <http://www.xanthir.com/b4Ko0>
- Seit CSS3 werden die Features mittels Modules definiert
 - Background & Borders 3: <https://drafts.csswg.org/css-backgrounds-4/>
 - FlexBox 1: <https://www.w3.org/TR/css-flexbox-1/>
 - Selector 4: <https://drafts.csswg.org/selectors/>
 - Oft als Synonym für CSS4 verwendet

Was ist CSS

- **Beim CSS können keine Versionen angegeben werden**
 - CSS wird vom Browser immer als neuste Variante interpretiert.
 - Selten relevant z.B. FlexBox
- **Browser Support für neue CSS Feature kann überprüft werden**
 - <http://caniuse.com/#feat=css-select3>
 - <http://caniuse.com/#feat=transforms3d>
- **CSS besteht aus Regeln welche das Aussehen des Dokumentes verändert**
- **CSS Regeln werden auf HTML / SVG angewendet**
- **CSS ist auch ein Standard**
 - <https://www.w3.org/Style/CSS/>

CSS DEFINIEREN

- Jedes HTML-Element hat ein Property «style»
- Styles können als Attributwert angegeben werden
- Verwendungszweck
 - Ausprobieren
 - Folien z.B. reveal.js
 - WYSIWYG-Editoren

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <span style="background: green; color: pink">Hello world</span>
</body>
</html>
```

Einschub Syntax: CSS Regel

```
span, div { background: blue; }
```

CSS-Regel besteht aus

■ Selektoren

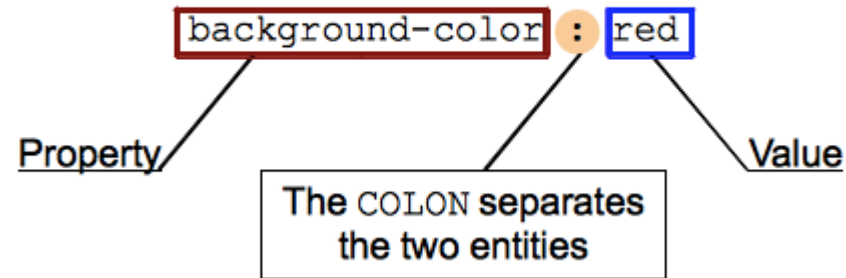
- z.B. für alle `span`'s und `div`'s im Dokument

■ Property : Value

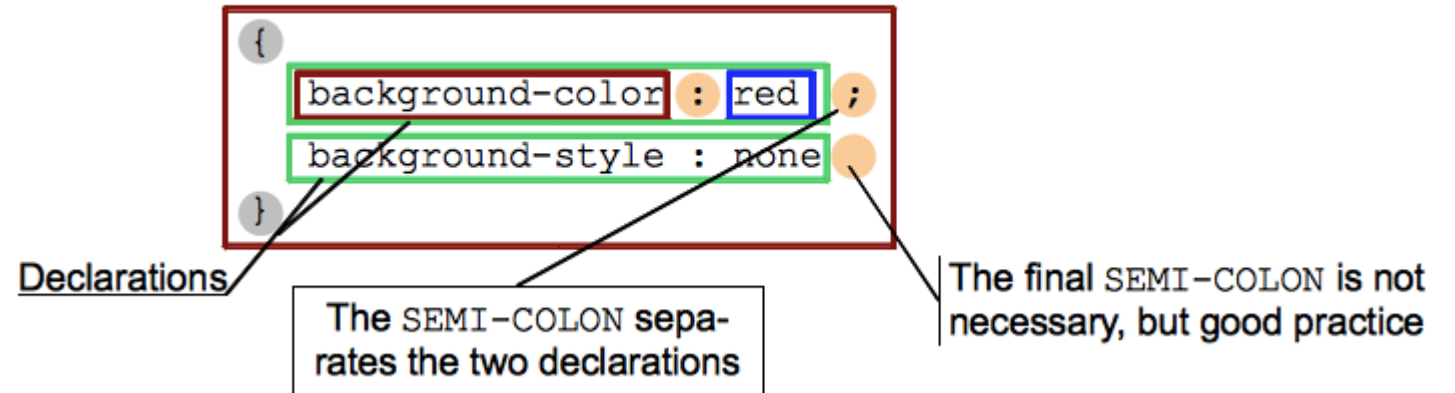
- z.B. setzt die Hintergrundfarbe für alle selektierten Elemente auf blau

Einschub Syntax: CSS Regel

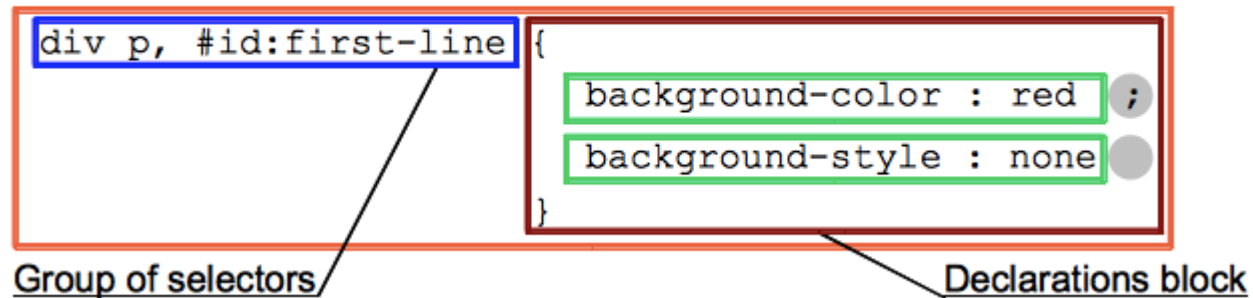
A CSS declaration :



A CSS declarations block:



A CSS ruleset (or rule):



<https://developer.mozilla.org/de/docs/Web/CSS/Syntax>

Einschub Syntax: CSS Kommentare

Kommentare wie folgt

```
/* multiline comment */
```

```
/*  
multiline comment  
*/
```

So nicht

```
// so keine CSS kommentare!
```

Einschub Syntax: Case sensitive

- Selektoren sind «Case-sensitive»
- Tipp: Immer klein schreiben

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    #Container{
      background: gray;
    }
  </style>
</head>
<body>
  <div id="container">
    <span>Hello world</span>
  </div>
</body>
</html>
```

Variante 2: Style

- Inhalt innerhalb von `<style> ... </style>` wird als Style interpretiert

- Im `<head>`

- Verwendungszweck

- Ausprobieren Advanced
- Vorlesung / Prüfungsaufgaben 😊
- Bessere Performance

- Ausblick: Diese Variante erlaubt **Scoped Styles**

- Sehr minimaler Support => aktuell nicht sinnvoll

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    span{
      background: red;
    }
  </style>
</head>
<body>
  <span>Hello world</span>
</body>
</html>
```

Repetition HTML-Tags: Style Tag

Content categories	Metadata content, and if the <code>scoped</code> attribute is present: flow content.
Permitted content	Text content matching the <code>type</code> attribute, that is <code>text/css</code> .
Tag omission	Neither tag is omissible.
Permitted parent elements	<p>If the scoped attribute is <i>not</i> present: where metadata content is expected or in a <code><noscript></code> element itself a child of <code><head></code> element.</p> <p>If the scoped attribute is present: where flow content is expected, but before any other flow content other than inter-element whitespace and <code><style></code> elements, and not as the child of an element whose content model is transparent.</p>
DOM interface	<code>HTMLStyleElement</code>

<https://developer.mozilla.org/de/docs/Web/HTML/Element/style>

<https://html.spec.whatwg.org/multipage/semantics.html#the-style-element>

Variante 3: File

- Mit dem [<link>](#) Tag kann ein externes CSS File geladen werden
 - [rel-attribute](#)
- Im `<head>`
- Verwendungszweck
 - Immer! (ausser man hat einen sehr guten Grund)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <link rel="stylesheet" href="file.css">
</head>
<body>
  <span>Hello world</span>
</body>
</html>
```

```
/*file.css:*/
span{
  background: blue;
  color: pink;
}
```


Variante 3a: File mit Screen

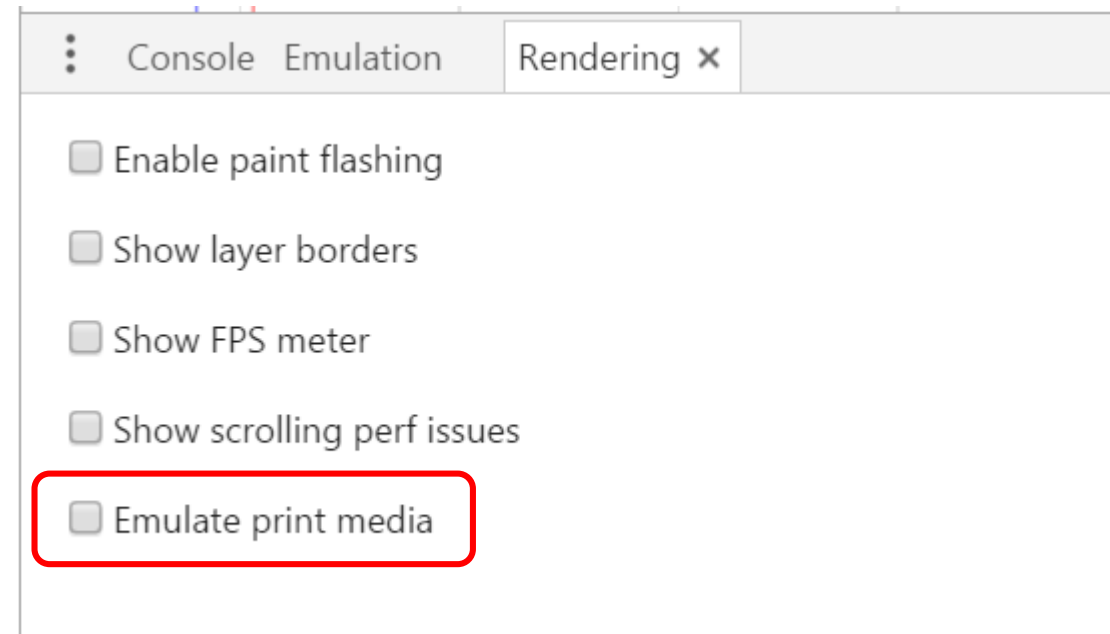
- Beim «link» Tag kann mit media=«print» bzw. «screen» angegeben werden, ob das Stylesheet für das jeweilige Medium angewendet werden soll.
- Beispiele
 - Werbungen beim Drucken ausblenden
 - Einblenden von einem speziellen Druck-Footer

```
<html lang="en">
<head>
  <link rel="stylesheet" href="print.css" media="print"/>
  <link rel="stylesheet" href="screen.css" media="screen"/>
</head>
<body>
  <p>Hello World</p>
</body>
</html>
```

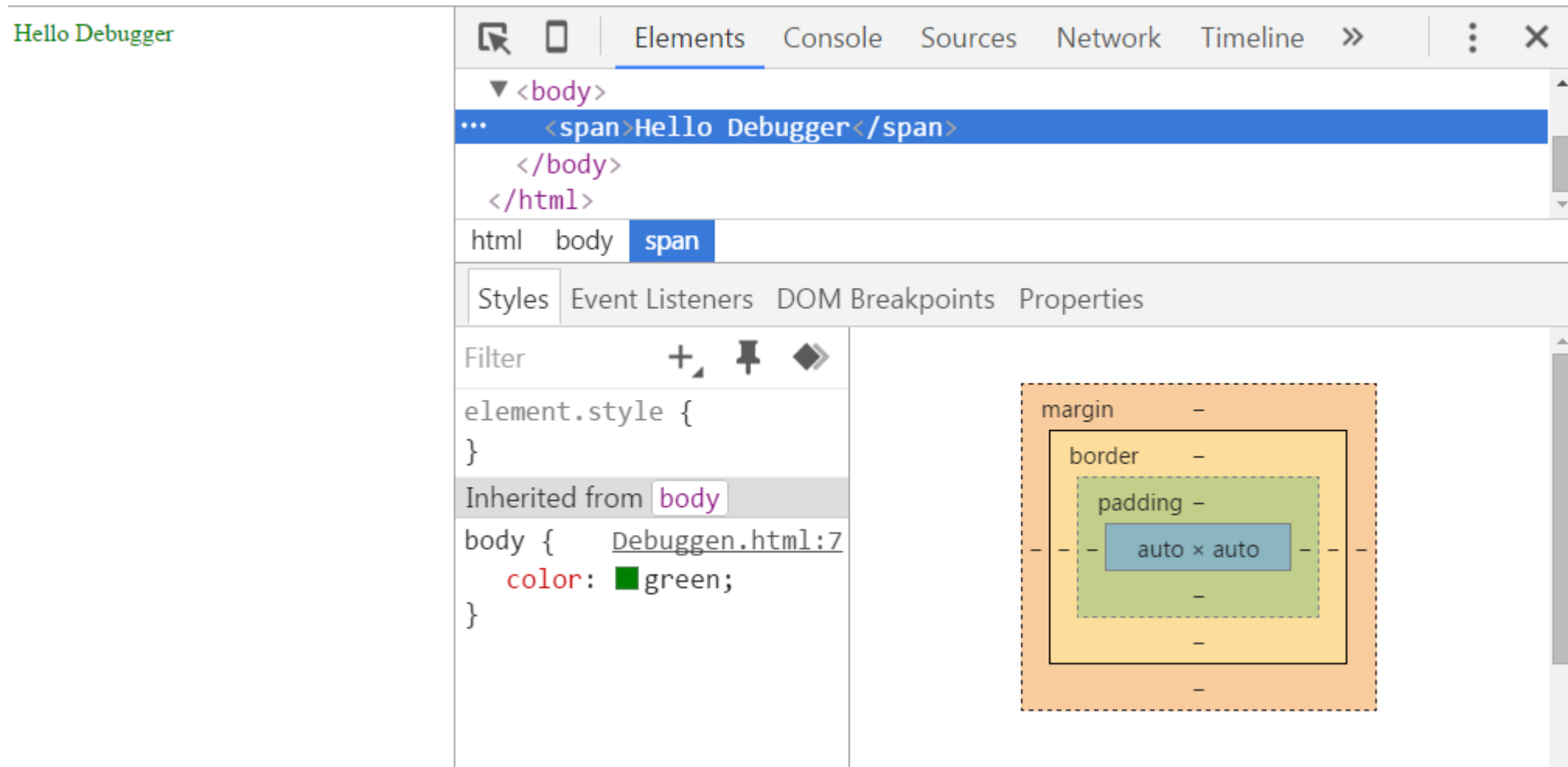
```
/*screen.css*/
p{
  color: blue;
}

/*print.css*/
p{
  color: red;
}
```

Die Chrome Dev. Tools können den «Print» Simulieren



CSS «DEBUGGEN»



SELEKTOREN BASICS

Der Selektor ist für die Selektion von Elementen verantwortlich. Auf diese Selektion werden die Styles angewandt.

Es gibt verschiedene Typen von Selektoren

Selektor-Typ	Beispiel
Typ	<code>a { }</code>
ID	<code>#container { }</code>
Universal	<code>* { }</code>
Klassen	<code>.info { }</code>
Attribute	<code>a[href] { }</code>
Pseudo-Elemente	<code>a::before { }</code>
Pseudo-Klasse	<code>tr:nth-child(2n){ }</code>

Selektoren können kombiniert werden

■ Selektiert alle HTML-Elemente vom angegebenen Type

- z.B. alle Links / alle Inputs / alle ...

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    a{
      color: red;
    }
  </style>
</head>
<body>
  <a href="#" target="_blank">Link1</a>
  <a href="#" target="_blank">Link2</a>
  <a href="#" target="_blank">Link3</a>
  <a href="#" target="_blank">Link4</a>
</body>
</html>
```

- ID Selektor wird mit #id erstellt
- Der performanteste Selektor
- Hinweis:
 - ID Selektoren sollten in modularem CSS nicht verwendet werden
 - Mehr dazu in WED2/3

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    #container {
      background: gray;
    }
  </style>
</head>
<body>
  <div id="container">
    <span>Hello world</span>
  </div>
</body>
</html>
```

KOMBINATOREN

■ Es gibt 4 verschiedene Varianten von Kombinatoren

■ Oft benutzt

- Kindselektor: $e > f$
- Nachfahrenselektor: $e f$

■ Für Spezialfälle

- Nachbarselektor: $e + f$
 - The lobotomized owl selector: $* + *$
 - Ausführlich: <http://alistapart.com/article/axiomatic-css-and-lobotomized-owls>
- Geschwisterselektor: $e \sim f$

■ Zum nachlesen: <https://wiki.selfhtml.org/wiki/CSS/Selektoren/Kombinator>

Kindselektor (Child Selector):

Werden zwei einfache Selektoren durch die schliessende spitze Klammer („>“) miteinander verbunden, z.B. „E > F“, so wird das Element F nur dann angesprochen, wenn es Kindelement eines E-Elements ist.

Nachfahrenselektor (Descendant Selector):

Werden zwei einfache Selektoren durch ein Leerzeichen („ “) miteinander verbunden, z.B. „E F“, so wird das Element F nur dann angesprochen, wenn es Nachfahre eines E-Elements ist.

```
<html lang="en">
<head>
  <style>
    body > ul { ①
      list-style-type: decimal;
    }
    body ul { ②
      opacity: 0.5;
    }
  </style>
</head>
<body>
  <ul> ①,②
    <li>
      <a href="#" target="_blank">AAAAAAA</a>
    </li>
    <ul> ②
      <li>
        <a href="#" target="_blank">C1</a>
      </li>
    </ul>
  </ul>
</body>
</html>
```

Kindselektor / Nachfahrenselektor

```
<html lang="en">
<head>
  <style>
    ol > li { ①
      text-decoration: line-through;
    }
    ol li { ②
      background: gray;
    }
  </style>
</head>
<body>
  <ol>
    <li> ①,2
      <a href="#" target="_blank">AAAAAAAAA</a>
    </li>
    <ul>
      <li> ②
        <a href="#" target="_blank">A1</a>
      </li>
    </ul>
    <li> ①,2
      <a href="#" target="_blank">BBBBBBBBBB</a>
    </li>
  </ol>
</body>
</html>
```


UNIVERSAL SELEKTOR

- Ein Universal Selektor besteht aus einem Sternzeichen „*“ (der Asterisk)
- Selektiert alle Elemente
- Wird oft für globale CSS-Resets gebraucht.
 - Später mehr dazu

```
* {  
  margin :0px;  
  padding: 0px;  
  box-sizing: border-box;  
}
```

- Kann bei Kombinatoren verwendet werden um in der HTML Strukturen weiter in die Tiefe zu navigieren z.B. Enkel selektieren

Universal

```
<html lang="en">
<head>
  <style>
    li > * { ①
      text-decoration: underline;
    }
    body * span { ②
      opacity: 0.5;
    }
  </style>
</head>
<body>
  <ul>
    <li>
      <span>A</span> ①,②
    </li>
    <li>
      <i>B</i> ②
    </li>
    <li>
      <strong>C</strong> ②
    </li>
  </ul>
  <span>D</span>
</body>
</html>
```

KLASSEN SELEKTOR

- **CSS Klasse hat nichts mit einer JavaScript bzw. Java Klasse zu tun!**

- **CSS definiert einen Klassen Selektor Type**

- **Dieser Type ist für Gruppierung von «gleichgesinnten» HTML-Elementen**

- **Ein Klassen Selektor wird mit einem . (Punkt) definiert z.B:**

- .info .alert .deprecated

- **Jedes HTML-Element hat ein Property «class»**

- Mehrere Klassen mit « » (Leerzeichen) getrennt

- **Kombinationen mit andern Selektoren sind möglich z.B:**

- *div.box* selektiert alle «div»s mit der Klasse «box»
- *.box.danger* selektiert nur Elemente, welche beide Klassen besitzen
- Best Practices: Neue CSS Klasse für diese Konstellation erstellen

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    .box {
      border : 1px black solid;
    }
    .box-danger{
      background : red;
    }
    .box-info{
      background : yellow;
    }
  </style>
</head>
<body>
  <div class="box box-danger">Warnung!</div>
  <div class="box box-info">Info!</div>
</body>
</html>
```

- Die Namen von CSS-Klassen sollten generisch gewählt sein.
- Die Klassen sollten so granular wie möglich sein.
 - z.B. Aufsplitten von box und box-error
 - Mehr in WED2/3
- In den meisten Fällen sind CSS Klassen ID-Selektoren / Typen vorzuziehen
 - Wiederverwendbarkeit
- Namen von Klassen sollten «generisch» sein und die den Sinn von der Klasse beschreiben und nicht das Aussehen z.B.:

```
/*schlecht*/  
.red{  
    background: red;  
}
```

```
/*oke*/  
.alert{  
    background: red;  
}
```

ATTRIBUTE SELEKTOR

- Seit CSS2 kann anhand von Attributen und Attributwerten selektiert werden.

- Überprüfen ob Attribute vorhanden ist:

- *[Attribute]
 - * ist optional
- h1[Attribute]
- .class[Attribute]
- #id[Attribute]

- Beispiel

```
a[href]{  
  color: green;  
}
```

■ Es können auch Werte überprüft werden

- [a=v]
 - Überprüft ob der Wert von Attribute gleich dem Wert v ist. **Komplett**
- [a~v]
 - Überprüft ob der Wert von Attribute das Wort v beinhaltet. **Alleinstehend**
- [a|=v]
 - Überprüft ob der Wert von Attribute mit dem Wort v startet. **Alleinstehend**
- [a^=v]
 - Überprüft ob der Wert von Attribute mit dem Wert v startet.
- [a\$=v]
 - Überprüft ob der Wert von Attribute mit dem Wert v endet.
- [a*=v]
 - Überprüft ob der Wert von Attribute das Wert v beinhaltet.

Attribute Selektor

```
<html>
<head>
<style>
  [type=submit] {font-weight: bold;}
  [title~=Berlin] {color: green;}
  [hreflang|=de] {font-style: italic;}
</style>
</head>
<body>

<p>
  <input type="text" value="Beispiel">
  <input type="submit" value="Absenden">
</p>

<ul>
  <li title="Berlin: eine schöne Stadt">Viele Informationen.</li>
  <li title="Die Stadt Berlin besteht aus mehreren Stadtteilen">Mehr Informationen.</li>
  <li title="Das Brandenburger Tor steht in Berlin">Noch mehr Informationen.</li>
</ul>

<ul>
  <li><a href="http://www.example.org/" hreflang="de">Beispielverweis</a></li>
  <li><a href="http://www.example.com/" hreflang="de-at">Beispielverweis</a></li>
  <li><a href="http://www.example.net/" hreflang="en-de">Beispielverweis</a></li>
</ul>
</body>
</html>
```

Diagramm zur Attribute Selektion:

- 1. `[type=submit]` (Attribut Selektor)
- 2. `[title~=Berlin]` (Partial Match Selektor)
- 3. `[hreflang|=de]` (Language Attribute Selektor)

Attribute Selektor

```
<html>
<head>
<title>CSS-Beispiel: Attributselektor</title>
  <style>
    a[href^="http://"] { font-weight: bold; ①
    a[href$=".pdf"]     { color: green;      ②
    a[href*="wiki"]    { font-style: italic; ③
  </style>
</head>
<body>
<h1>Teilübereinstimmungen</h1>
<ul>
  <li>
    <a href="http://example.com/">Beispiellink</a> ①
  </li>
  <li>
    <a href="https://example.com/example.pdf">Ein PDF-Dokument</a> ②
  </li>
  <li>
    <a href="http://wiki.selfhtml.org/">Das selfhtml wiki</a> ①,3
  </li>
  <li>
    <a href="https://example.org/WIKI/example.pdf/">Ein PDF-Dokument</a>
  </li>
</ul>
</body>
</html>
```

PSEUDO ELEMENT

- Frage: Weshalb kann folgendes Resultat nicht mit den bis jetzt gelernten Selektoren erreicht werden?

```
<div class="container">  
  <span>Das ist eine kurze Zeile</span>  
  <span>Das ist eine kurze Zeile</span>  
  <span>Das ist eine kurze Zeile</span>  
  <p>Das ist eine sehr sehr lange Zeile....</p>  
  <p>Das ist eine kurze Zeile</p>  
  <p>Das ist eine kurze Zeile</p>  
</div>
```

Das ist eine kurze Zeile Das ist eine kurze Zeile
Das ist eine kurze Zeile

Das ist eine sehr sehr lange Zeile. Das ist eine sehr
sehr lange Zeile. Das ist eine sehr sehr lange Zeile.
Das ist eine sehr sehr lange Zeile. Das ist eine sehr
sehr lange Zeile. Das ist eine sehr sehr lange Zeile.
Das ist eine sehr sehr lange Zeile.

Das ist eine kurze Zeile

Das ist eine kurze Zeile

- Pseudoelemente erlauben es bestimmte Teile eines Elementes zu formatieren.

- **::first-line**

- Selektiert die erste Textzeile eines Elementes

- **::first-letter**

- Selektiert den ersten Buchstabe eines Elementes

- **Zeigt nur Wirkung bei Block- und blockähnlichen Elementen**

- Inline-block
 - Listenpunkte
 - Tabellenüberschriften und –zellen

```
<style>

p::first-line{
    background: gray;
}
.container::first-line{
    background: red;
}
.container ::first-letter{
    font-weight: bold;
}
span{
    display: inline-block;
}

</style>
```

::before ::after

- **::before** und **::after** erzeugen ein neues Element, welches vor bzw. nach dem selektiertem Element eingefügt wird.

```
<html lang="en">
<head>
<meta charset="UTF-8">
<style>
  .question::before{
    font-weight: bold;
    content: '¿';
  }

  .question::after{
    font-weight: bold;
    content: '?';
  }
</style>
</head>
<body>
  <p class="question">Te gusta el verano</p>
</body>
</html>
```

Resultat

¿Te gusta el verano?

Debugger

```
<p class="question">
  ::before
  "Te gusta el verano"
  ::after
</p>
```


PSEUDO KLASSEN

- **Pseudoklassen sind Selektoren, welche Eigenschaften / Zustände von Elemente Berücksichtigen**
- **Liste aller Pseudoklassen:** <https://wiki.selfhtml.org/wiki/CSS/Selektoren/Pseudoklasse>
- **Es wird unterschieden**
 - strukturelle Pseudoklassen
 - :first-child, :nth-child(), :empty,...
 - dynamische Pseudoklassen
 - :hover, :active, :focus, :visited,...
 - diverse
 - :lang(), :not(), :matches()
- **Die Pseudoklassen sind Teil der Übungen / Selbststudium.**

■ Einige Pseudoklassen können mit einem «n» umgehen z.B. :nth-child()

- :nth-child(1)
 - Selektiert das erste Element (index 1)
- :nth-child(2)
 - Selektiert das zweite Element (index 2)
- :nth-child(2n)
 - Selektiert alle Elemente mit $\text{index} \% 2 == 0$
- :nth-child(2n+1)
 - Selektiert alle Elemente mit $\text{index} + 1 \% 2 == 0$
- :nth-child(3n)
 - Selektiert alle Elemente mit $\text{index} \% 3 == 0$
- :nth-child(odd) bzw. :nth-child(even)
 - Selektiert alle geraden bzw. ungeraden Elemente

■ Index startet bei 1!



:nth-child() Tabelle Formatieren

■ Abwechselnde Hintergrundfarbe verbessert die Lesbarkeit einer Tabelle

```
table tr:nth-child(2n) {  
    background: gray;  
}
```

A	B	X
A	1	X
C	2	X
D	3	X
E	4	X
F	5	X
G	6	X

SELEKTOR HINTERGRUND INFOS

■ Browser interpretiert ein CSS-Selektor von rechts nach links

```
#id ul li a { color: red; }
```

■ Alle «a» Elemente werden selektiert

- Für jedes Element wird nach einem Parent-«li»-Tag gesucht.
 - Für jedes Element wird nach einem Parent-«ul»-Tag gesucht.
 - Für jedes Element wird nach einem Parent mit der #id gesucht.
 - Style wird angewendet
- Falls die Suche fehlschlägt – wird die Regel nicht angewandt

■ Wie wird folgender Ausdruck interpretiert?

```
[href] {  
  color: red;  
}
```

https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Writing_efficient_CSS

Wichtigste Punkte

■ Avoid the descendant selector

- A > B wenn möglich A B vorziehen

■ Don't qualify ID rules with tag names or classes

- A > #ID ist unnötig da #ID eindeutig ist

■ Rely on inheritance

```
/*bad*/  
.navigation li a { font-family: Georgia, Serif; }  
/*besser*/  
.navigation { font-family: Georgia, Serif; }
```

■ Avoid universal rules

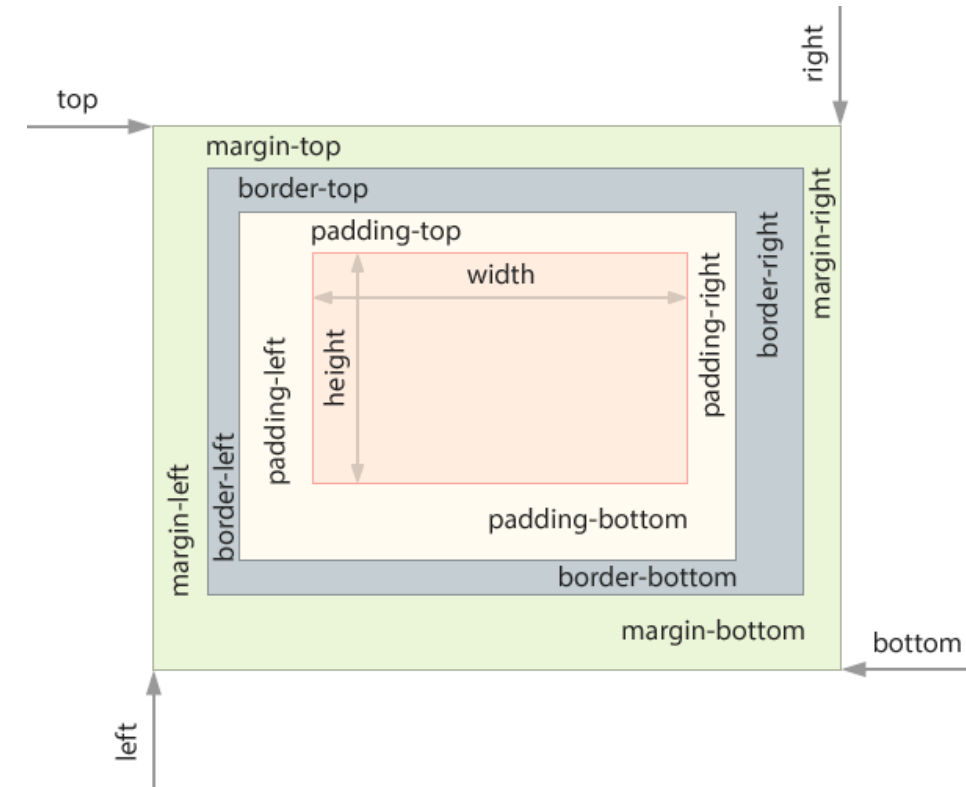
- Auch folgendes ist eine Universal Regel:

```
[href] { }
```

BOX-MODEL

Box-Model

- **Jedes Element besteht aus**
 - Inhalt (content)
 - Innenabstand (padding)
 - Rahmen (border)
 - Aussenabstand (margin)
- **Jeder dieser Punkte erzeugt eine Box, die von der Äusseren umschlossen wird.**
- **Die Grösse des Elementes ist die Summe von Content + Padding + Border + Margin**
- **Background wird auf content + padding angewendet**
- **Margin kann auch negativ sein.**

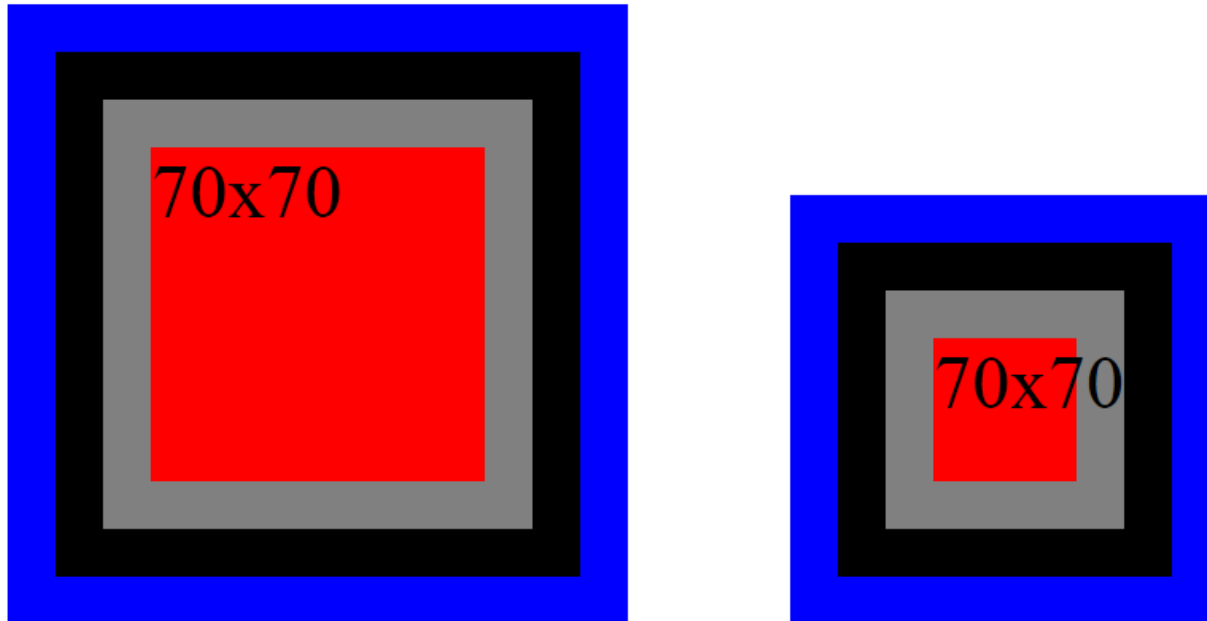


- **«Box-Model-Simulator»:** <http://codepen.io/carolineartz/full/ogVXZj>

Box-sizing

`box-sizing: border-box;`

Mit dieser Definition wird Padding / Border vom Content abgezogen.



Wichtig für Fluides Design.

BORDER / OUTLINE

■ Border

- Breite des Rahmens
- Jede Kante kann einzeln formatiert werden
- Abrunden möglich
- Bild als Rahmen-Hintergrund verwendbar
- Border wird der Grösse des Elementes hinzugefügt

■ Outline

- Outline wird **nicht** der Grösse des Elementes hinzugefügt
- Gewisse Elemente ignorieren den Border (z.B. CheckBox) Outline funktioniert
- Viel weniger Optionen als Border

CSS EIGENSCHAFTEN

Einige CSS Properties können in einem Property zusammengefasst werden.

■ Margin (& Padding)

■ `margin: 1px`

- `margin-top: 1px;`
- `margin-right: 1px;`
- `margin-left: 1px;`
- `margin-bottom: 1px;`

■ Weitere Varianten

- `margin: 1px 2px 3px 4px` (top | right | bottom | left – Uhrzeigersinn)
- ~~`margin: 10px 20px` (vertikal | horizontal)~~
- ~~`margin: 10px 5px 1px` (top | horizontal | bottom)~~

■ Kombinierte Properties sind zu empfehlen, wenn Sie die Übersicht verbessern.

- `margin: 1px 0 0 0 => margin-top: 1px;`

■ Border

■ `border: 1px solid black`

- `border-width: 1px;`
 - `border-top: 1px;`
 - `border-right: 1px;`
 - `border-left: 1px;`
 - `border-bottom: 1px;`
- `border-color: black;`
- `border-style: solid;`

■ `border-bottom: 1px solid black`

- `border-bottom-color: 1px;`
- `border-bottom-style: solid;`
- `border-bottom-width: black;`

Vererbte Properties

Einige CSS Properties werden vom Parent- zum Child-Element weitergegeben

Beispiele:

- Font, Color, Schriftgrösse

Bei der MDN wird bei den Properties angegeben ob sie «Inherited» sind. Z.B.

- <https://developer.mozilla.org/en-US/docs/Web/CSS/color>

Im Browser sichtbar:

```
Inherited from body  
body {  
  color: red red;  
}
```

- Inherited Properties ausnutzen und nicht auf jedem Child z.B. den Font setzen, sondern auf dem Container.

Default Werte

- **Browser definiert ein default Set von sinnvollen Element-Styles.**
 - So wird für den body tag immer 8px margin definiert.
- **Die meisten Projekte nutzen «Style-Resets» um Default Styles abzuschalten.**
- **Einfachste Variante:**

```
* {  
  padding:0;  
  margin:0;  
}
```



■ Komplexere:

- <http://yuilibrary.com/yui/docs/cssreset/>
- <https://perishablepress.com/a-killer-collection-of-global-css-reset-styles/>
- <http://meyerweb.com/eric/tools/css/reset/>

■ Wichtig

- Es werden alle Styles entfernt
 - , <a>, <i>, ..
- Es muss sicher gestellt sein, dass diese Tags wieder sinnvoll «gestyled» werden.

■ Wie wird ein div formatiert, falls die Funktionen rgb / rgba nicht unterstützt sind?

```
div{  
  color: #FF0000;  
  color: rgb(255,0,0);  
  color: rgba(255,0,0,0.1);  
  color: myfunnycolor;  
}
```

■ Nachfolgende CSS Deklaration überschreiben vorhergehende...

■ ...nicht wenn diese Zeile «fehlerhaft» ist bzw. nicht unterstützt wird. In diesem Fall wird die Zeile ignoriert.

■ Viele Browser Hersteller haben für «eigene» CSS Attribute ein Prefix

- -webkit- (Chrome, newer versions of Opera.)
- -moz- (Firefox)
- -o- (Old versions of Opera)
- -ms- (Internet Explorer)

■ Grund: Experimentelle Styles und / oder noch nicht standardisiert.

■ Beispiel:

Achtung!

Die Eigenschaft `filter` ist derzeit (April 2015) nur in die Browser Firefox und IE Edge 13 implementiert, deshalb muss man **proprietäre Eigenschaften** verwenden.

für Android, Chrome, Opera und Safari:

- `-webkit-filter`

■ Für Rückwärtskompatibilität werden oft alle Variationen angegeben. Hier helfen Prefixer Generator oder CSS Preprozessoren

- Mehr in WED2

■ Der «Standard» immer als letztes angegeben

```
#container {  
  display : -webkit-box;  
  display : -webkit-flex;  
  display : -ms-flexbox;  
  display : flex;  
}
```

DISPLAY

■ **Inline** z.B. span a

- Erlaubt left & right margin und padding aber nicht top und bottom
- Ignoriert “width” und “height”
- Erlaubt andere Elemente auf der gleichen Zeile
- Bricht die Zeile – falls ungenügend Platz vorhanden ist – wie text um.
- Nimmt nur den notwendigen Platz ein.
- Erzeugt einen “Whitespace”!
 - Mehr in der DOM Vorlesung

■ **Block** z.B. h1 div form p

- Erlaubt margin / padding
- Erzeugt einen Linienumbruch
- Füllt die ganze Zeile

■ **Inline-Block** z.B. Inline-Flex Inline-Table

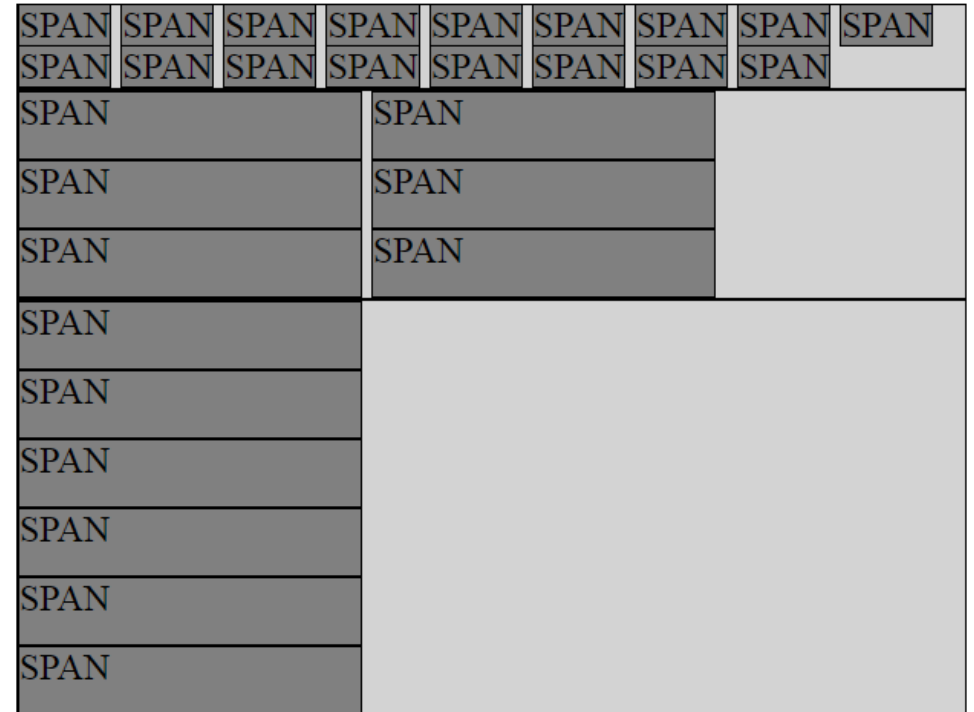
- Erlaubt margin / padding
- Erlaubt andere Elemente auf der gleichen Zeile
- Erlaubt “width” und “height”
- Nimmt nur den notwendigen Platz ein.
- Erzeugt einen “Whitespace”!

■ Details: <https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Anzeige/display>

Display Attribute

- Das display Attribute erlaubt es den Type zu ändern.
- Z.B. ein span als Inline/ Block-Inline / Block

```
.container > span{
  border: 1px solid black;
  height: 30px;
  width: 150px;
  background: gray;
  box-sizing: border-box;
}
.inline > span{
  display: inline;
}
.inline-block > span{
  display: inline-block;
}
.block > span{
  display: block;
}
div{
  overflow: hidden;
}
```



Display: none / Visibility : none

■ Display:none

- Das Element wird aus dem «Fluss» entfernt
- Wird von Screenreader nicht mehr gefunden. (gewollt)

■ Visibility:hidden

- Der Platz vom Element wird reserviert aber als «leer» gezeichnet
- Wird von Screenreader nicht mehr gefunden. (gewollt)

■ Opacity: 0

- Gleicher effekt wie **Visibility:hidden**
- Aber vom Screenreader erkennbar

■ Visibility:collapse

- Wie display:none aber nur für Table-Elemente
- Display:none ist vorzuziehen. Besserer Support

POSITION

- **HTML-Elemente werden nach Typ (Block, Inline, ...) von links nach rechts und von oben nach unten platziert.**
- **In speziellen Szenarios sollten HTML-Elemente übereinander liegen**
- **Beispiel**
 - Pop Up
 - Layers
- **Mit «Position» kann ein Element aus dem Fluss genommen werden**
- **Wichtig: Position nur verwenden, wenn's nötig ist. Überprüfen ob es überall funktioniert!**



■ Absolute

- Kann genutzt werden um ein beliebiges Element absolut zu definieren mit
 - top, left, bottom, right
- Werte sind relativ zum ersten Parent mit «absolute» oder «relative» Position
- Kann genutzt werden um ein Element über andere Elemente zu legen
 - z.B. Fotobuch

■ Fixed

- Kann genutzt werden um ein Element fix an einem Ort auf dem Screen zu platzieren.
 - z.B. Menü welches immer sichtbar sein sollte

■ Bei Absolute und Fixed positionierten Elementen wird der Platz nicht reserviert.

■ Relative

- Der Platz wird reserviert.
- Der Platz ist im «Fluss».
- Kann sich selbst auch «absolute» positionieren – verlässt dann aber sein «reservierten» Platz.
 - Die Werte sind relativ zum reservierten Platz.
- Oft als Parent für absolut positionierten Inhalt definiert.

■ Static

- Default

FLOAT

- Kann verwendet werden um Text um ein Bild fließen zu lassen.



Lorem ipsum dolor sit amet, consetetur sadipscing
dolore magna aliquyam erat, sed diam voluptua. A
clita kasd gubergren, no sea takimata sanctus est L
consetetur sadipscing elitr, sed diam nonumy eirm
sed diam voluptua. At vero eos et accusam et justo
takimata sanctus est Lorem ipsum dolor sit amet. I
diam nonumy eirmod tempor invidunt ut labore et
accusam et justo duo dolores et ea rebum. Stet clita
dolor sit amet. Lorem ipsum dolor sit amet, conset
ut labore et dolore magna aliquyam erat, sed diam
rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.
diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.
diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

- Hinweis: Floats wurden / werden oft für Layouten verwendet. Sollte heute vermieden werden
 - Mehr im WED2

CSS KASKADE

- **Mehrere Stylesheets können das gleiche Dokument stylen.**
- **Mehrere widersprechende Styles sind möglich**
- **Die Kaskade ist ein «Algorithmus» um die unterschiedlichen Style-Quellen zu verschmelzen.**
- **Die Kaskade setzt sich aus folgenden Komponenten zusammen**
 - Aus welcher Quelle kommt der Style
 - Spezifität des Selektors
 - Reihenfolge der Style
 - Wird !important verwendet

■ Stylesheets haben unterschiedliche Quellen

1. Browser (kleinste Priorität)
 - Default Werte
2. Benutzer
 - Vom Nutzer vom Browser getätigte Einstellungen.
 - Wird sehr selten eingesetzt
3. Autor (höchste Priorität)
 - StyleSheets, welche mit der Webseite mitgeliefert werden.

- Falls Styles den gleichen Selektor haben, werden die Werte von der zuletzt definierten CSS Regel übernommen.

```
h1{  
  font-weight: bold;  
  font-size: 20px;  
}  
h1{  
  font-size: 30px;  
}
```

```
h1 { Rheinforme.html:12  
  font-size: 30px;  
}  
h1 { Rheinforme.html:7  
  font-weight: bold;  
  font-size: 20px;  
}
```

- **CSS Regeln werden nach ihrer Spezifität gewichtet. Die Spezifität wird aus dem Selektor berechnet.**
- Für jeden Selektor werden drei Zähler (A, B und C) mit dem Startwert Null festgelegt. Jeder Bestandteil eines Selektors wirkt sich auf diese Zähler aus
 - Der Zähler A wird durch jedes Vorkommen eines ID-Selektors um Eins erhöht.
 - Der Zähler B wird durch jedes Vorkommen eines Attribut- oder Klassenselektors bzw. einer Pseudoklasse um Eins erhöht.
 - Der Zähler C wird durch jedes Vorkommen eines Typselektors oder eines Pseudoelements um Eins erhöht.
- Der Universalselektor verhält sich neutral, er wird ignoriert.
- Die Pseudoklasse :not() selbst wird ignoriert, die Selektoren innerhalb der Klasse werden jedoch wie vorbeschrieben gewertet.

Beispiel: - Spezifität von Selektoren von gering nach hoch geordnet

```
*                /* A=0, B=0, C=0, Spezifität 0 0 0 */
h1               /* A=0, B=0, C=1, Spezifität 0 0 1 */
ul li           /* A=0, B=0, C=2, Spezifität 0 0 2 */
a::after        /* A=0, B=0, C=2, Spezifität 0 0 2 */
p:first-child   /* A=0, B=1, C=1, Spezifität 0 1 1 */
a:not([href])   /* A=0, B=1, C=1, Spezifität 0 1 1 */
ul.nav [href]   /* A=0, B=2, C=1, Spezifität 0 2 1 */
#author         /* A=1, B=0, C=0, Spezifität 1 0 0 */
#editor p       /* A=1, B=0, C=1, Spezifität 1 0 1 */
```

Spezifität des style-Attributes

Werden Eigenschaften in einem style-Attribut festgelegt, so ist diese Eigenschaft spezifischer als jeder Regelsatz in einem Stylesheet. Die Errechnung der Spezifität erfolgt mit Hilfe eines vierten Zählers (hier A), der sich um Eins erhöht, wenn ein style-Attribut gesetzt wurde.

Beispiel: - Berechnung der Spezifität mit Berücksichtigung des style-Attributs

<code>*</code>	<code>/* A=0, B=0, C=0, D=0, Spezifität 0 0 0 0 */</code>
<code>h1</code>	<code>/* A=0, B=0, C=0, D=1, Spezifität 0 0 0 1 */</code>
<code>ul li</code>	<code>/* A=0, B=0, C=0, D=2, Spezifität 0 0 0 2 */</code>
<code>a::after</code>	<code>/* A=0, B=0, C=0, D=2, Spezifität 0 0 0 2 */</code>
<code>p:first-child</code>	<code>/* A=0, B=0, C=1, D=1, Spezifität 0 0 1 1 */</code>
<code>a:not([href])</code>	<code>/* A=0, B=0, C=1, D=1, Spezifität 0 0 1 1 */</code>
<code>ul.nav [href]</code>	<code>/* A=0, B=0, C=2, D=1, Spezifität 0 0 2 1 */</code>
<code>#author</code>	<code>/* A=0, B=1, C=0, D=0, Spezifität 0 1 0 0 */</code>
<code>#editor p</code>	<code>/* A=0, B=1, C=0, D=1, Spezifität 0 1 0 1 */</code>
<code>style=""</code>	<code>/* A=1, B=0, C=0, D=0, Spezifität 1 0 0 0 */</code>

!important

- Mit !important kann ein Style als wichtig markiert werden.
- Sollte nie verwendet werden falls das CSS unter eigener Kontrolle ist.
- Nützlich falls ein Framework/CMS Styles setzt und !important die einzige Möglichkeit ist diese Styles zu überschreiben.

```
h1{  
  font-weight: bold;  
  font-size: 20px; !important;  
}  
h1{  
  font-size: 30px;  
}
```

```
h1 {                                important.html:12  
  font-size: 30px;  
}  
  
h1 {                                important.html:7  
  font-weight: bold;  
  font-size: 20px !important;  
}
```

Die relevanten Regeln sind nach folgenden Kriterien sortiert:

- 1. Deklarationen im Browser-Stylesheet (kleinste Priorität)**
 - 2. Deklarationen des Benutzers**
 - 3. Deklarationen des Autors**
 - 4. Wichtige Deklarationen (mit !important) des Autors**
 - 5. Wichtige Deklarationen (mit !important) des Benutzers (grösste Priorität)**
 - Der Benutzer soll das letzte Wort haben bei Styles z.B. Grösse Schriftgrösse
- Innerhalb von jedem Punkt wird nach der Spezifität sortiert
 - Bei gleicher Spezifität gewinnt die Reihenfolge

EINHEITEN

■ px

- Ist die Basis-Einheit vom Browser
- Ist eine virtuelle Grösse
- Mehr in WED2 ☺

■ em

- Relativ zur Parent-Schriftgrösse

■ rem

- Relativ zur Root-Element (html)

■ vw / vh

- %-Grösse vom Viewport

FONT & COLOR

font-family

```
body{  
  font-family: Georgia, 'Times New Roman', serif;  
}
```

- Ist eine Liste von Schriftarten
- Die erste vom Browser unterstützte Schriftart wird gewählt
- Letzte Schriftart in der Liste sollte eine generische sein
 - serif, eine Schriftart mit Serifen
 - sans-serif, eine Schriftart ohne Serifen
 - cursive, eine Schriftart für Schreibschrift
 - fantasy, eine Schriftart für ungewöhnliche Schrift
 - monospace, eine Schriftart mit dicktengleichen Zeichen
- Die Schriftarten sollten ähnlich sein um allfällige Layout Probleme vorzubeugen. Z.B. gleiche Dimensionen besitzen
 - Job vom Designer 😊

■ font-style

- normal, normaler Schriftstil, **Default**
- italic, kursiver Schriftstil
- oblique, schräg gestellter Schriftstil

■ font-variant

- normal, normale Schriftvariante, **Default**
- small-caps, Kapitälchen

■ font-size

- Darstellungsgrösse der Schrift
- Basis für relative Einheiten wie rem und em
- Auch relative Einheiten möglich
- Wichtig: font-size werden vererbt
 - Bei Verschachtelungen von relativen Einheiten wird die Schrift bei jeder Hierarchiestufe kleiner

■ font-weight

- normal, normale Strichstärke
- bold, fett
- 100,200,300,400,500,600,700,800,900, extra-dünn (100) bis extrafett (900)
- lighter, dünner als im Elternelement
- bolder, fetter als im Elternelement

■ line-height

- Kann genutzt werden um die Zeilen Höhe zu verändern
- Üblicherweise mit relativen Einheiten z.B. 1.4em


■ text-decoration

- underline, unterstrichen
- overline, „überstrichen“
- line-through, durchgestrichen

- **color ist das Attribute für die Schriftfarbe oder im Allgemeinen für den «Vordergrund»**
 - Fixe Werte: red, blue, ...
 - Hex Werte: #RRGGBB (Rot Grün Blau)
 - Funktionen: rgb(255, 0, 0) (Rot, Grün, Blau)
- **Transparente Werte möglich**
 - Funktion: rgba(230,100,100,0.1) (Rot Grün Blau Alpha)

BESPIELE

Webshop

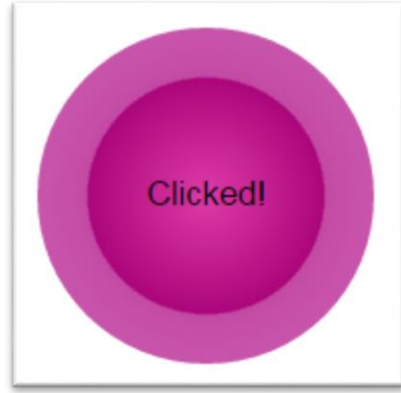
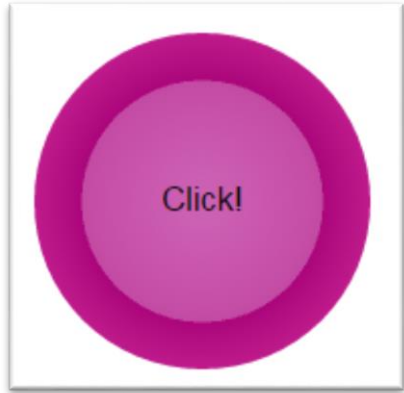
 Warenkorb

 Wetter

 Kontakt

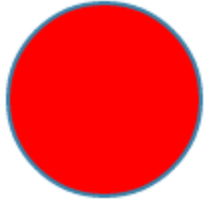
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilis. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim

Fancy Button



```
<body>  
  <button></button>  
</body>
```

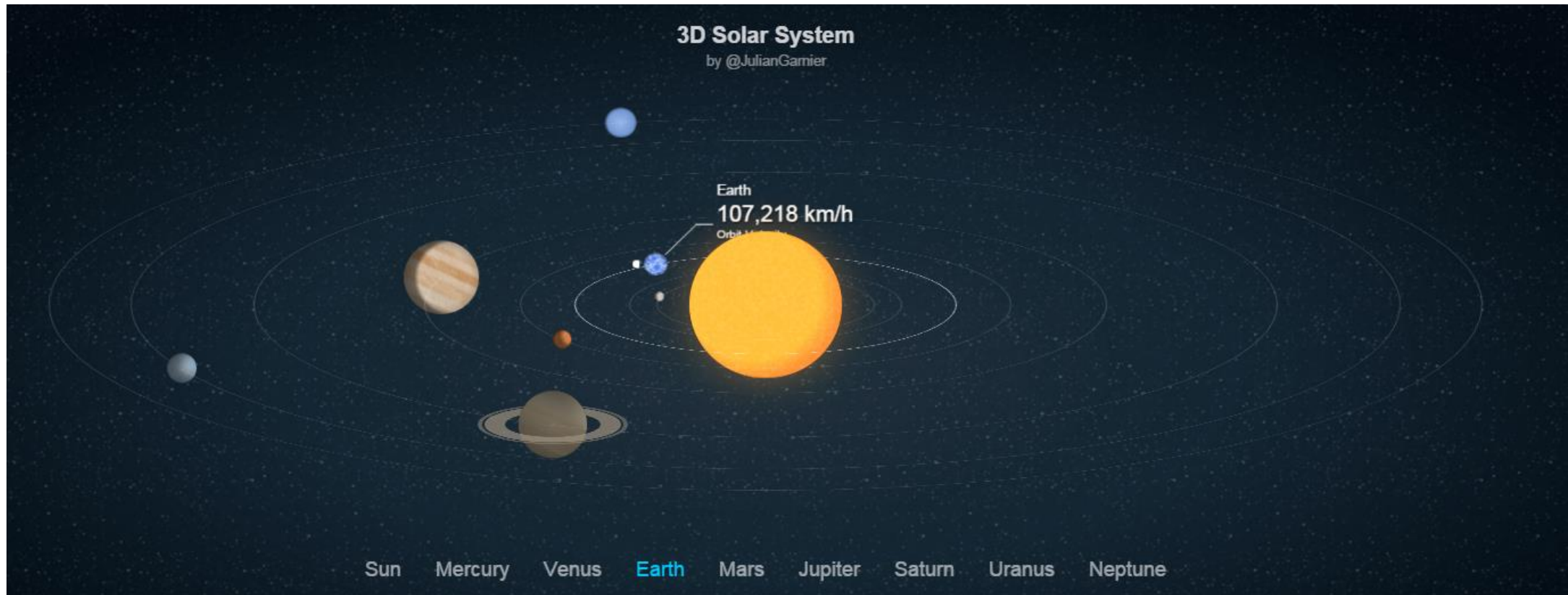
SVG Stylen



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    #circle1{
      fill : red;
    }
    #circle1:hover {
      fill: blue;
    }
  </style>
</head>
<body>
<svg >
  <circle id="circle1" cx="50" cy="50" r="48" fill="blue" stroke="#3983ab" stroke-width="2"/>
</svg>
</body>
</html>
```

CSS Solar System

<https://codepen.io/juliangarnier/pen/idhuG>



QUELLEN / ZUM NACHSCHLAGEN

- <https://wiki.selfhtml.org/>
- <https://developer.mozilla.org/de/>

AUSBLICK WED2

- Responsive Design
- Layouting
- SASS
- Animationen
- Mehr CSS! 😊

