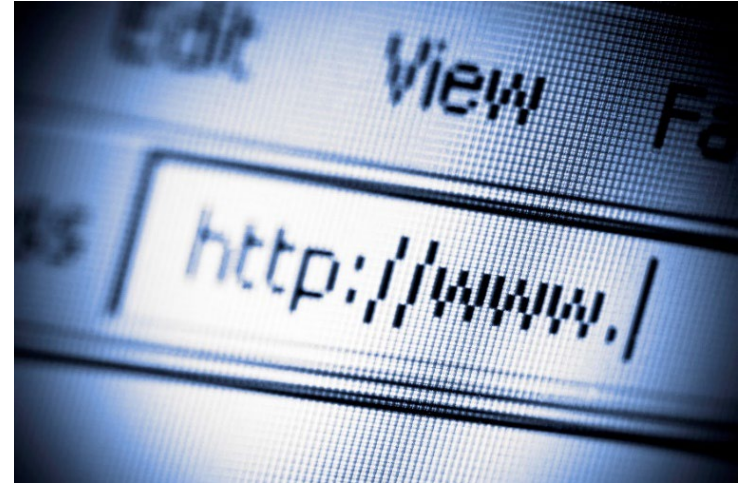


Web-Technologien

CSS-Grundlagen

- CSS-Syntax
- CSS-Selektoren und Kombinatoren
- CSS-Kaskadierung
- Inline- und Block-Elemente, Box-Modell

Letzte Aktualisierung: 30. September 2022



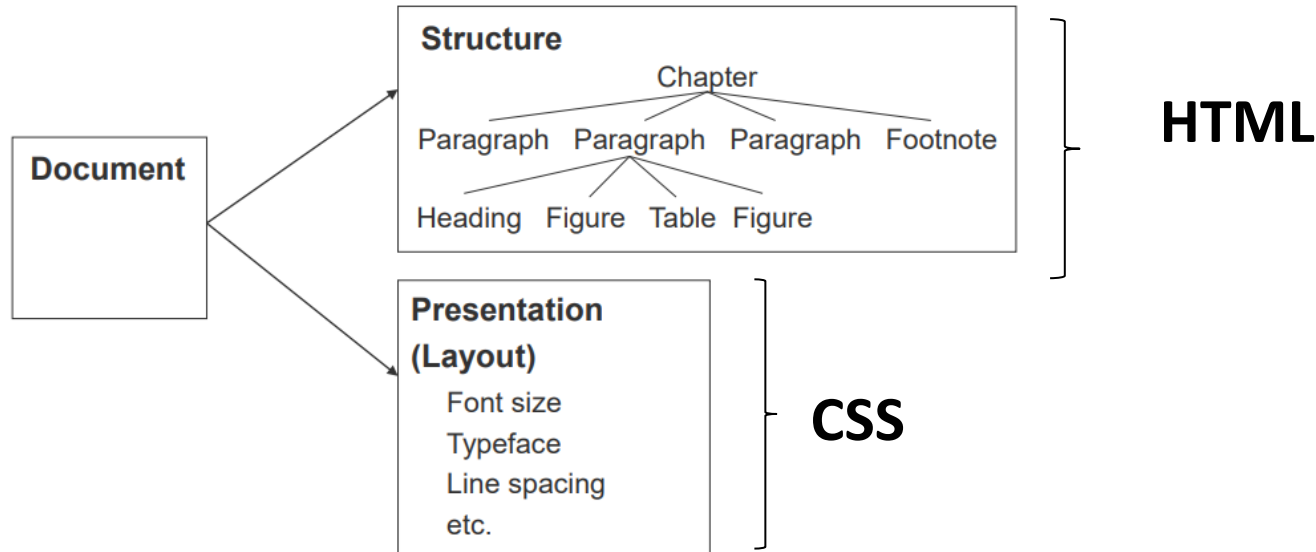
Lernziele

- Verstehen wie mittels CSS die **Darstellung** der durch HTML strukturierten Elemente geschieht.
- Wissen wie CSS **syntaktisch aufgebaut** ist und wie damit die Darstellung beeinflusst werden kann.
- Wissen wie CSS-Selektoren und Kombinatoren funktionieren (Tags, IDs, Klassen, etc.).
- Wissen wie mittels Inline- und Blockelementen Teilbereiche formatiert werden können und wie mit dem Box-Modell Blockelemente beeinflusst werden können.

Cascading Style Sheets: Einführung

Idee: Struktur und Präsentation trennen.

- Dokumentstruktur → Logische Struktur eines Dokuments
- Dokumentpräsentation → Formatierung und Erscheinungsbild



Cascading Style Sheets: Einführung

Beispiel: Zeitung

- Aussehen der Zeitung (Corporate Design) wird einmal festgelegt → **Design** (CSS).
- Reporter schreibt → **Inhalt** (HTML).
- Redakteur baut Spalten, Absätze und Bilder sinnvoll zusammen → **Struktur** (HTML).

Cascading Style Sheets: Einführung

Beispiel: studivz.net (mit CSS)

The screenshot shows the login page of studivz.net with a red header and a sidebar. The main content area features a large image of a couple with the text "Erst „Twilight“, jetzt „Vampire Diaries“! Muss das sein?". Below this is a "Kostenlos registrieren" button. The login form includes fields for "Dein Vorname", "Dein Nachname", "Geburtsdatum" (Tag, Monat, Jahr), "Geschlecht" (weiblich, männlich), and "Was ist das hier?". A "Weiter" button is at the bottom of the form. The sidebar contains a "Passwort vergessen?" link, a "Sitzung sichern" checkbox, and a "Plattenteller" link. The footer contains various links and logos.

studivz.net (ohne CSS)

The screenshot shows the login page of studivz.net without CSS styling. The layout is less structured, with the login form and sidebar elements appearing more basic. The "Kostenlos registrieren" button is still present. The login form includes fields for "E-Mail" and "Passwort", a "Passwort vergessen?" link, a "Sitzung sichern" checkbox, and a "Plattenteller" link. The footer contains various links and logos.

[Aktivierungslink noch einmal zusenden](#)

Es wird getanzt, was auf den [Plattenteller](#) kommt!

Studentenverzeichnis / studivZ

- [Immatrikulieren](#)
- [Hilfe](#)
- [Handy](#)
- [Blog](#)

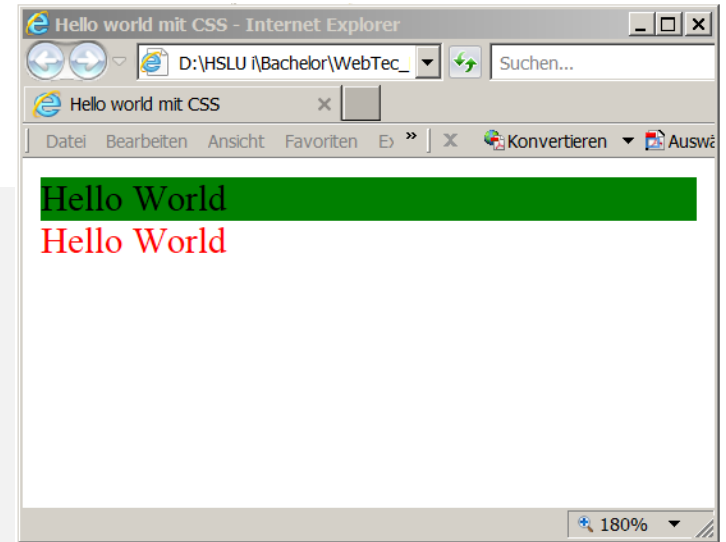


CSS-Historie in Kürze

- Version 1.0 im Jahr 1996 vorgestellt.
- Seit 1998 ist Version 2.0 Standard.
- Version 3.0 ist fertig, es wird aber immer noch daran gearbeitet:
<http://www.w3.org/Style/CSS/current-work> .
- Nach wie vor werden Module entwickelt und standardisiert.

«Hello World» mit CSS

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello world mit CSS</title>
  <style>
    p { background-color: green }
    #hello { color: red; }
  </style>
</head>
<body>
  <p>Hello World</p>
  <div id="hello">Hello World</div>
</body>
</html>
```



CSS-Prinzip

HTML-Elemente werden **selektiert** und dieser Selektion wird ein **Style zugeordnet**.

Grundlegende Syntax:

```
selector {  
    property1: value1;  
    property2: value2;  
    ...  
}
```

Beispiel:

```
h1 {  
    font-size: 16px;  
    color: red;  
}
```



Setzt für alle H1-Elemente im HTML-Dokument die Schriftgrösse auf 16px und die Farbe auf rot.

Zwei Beispiele mit CSS

CSS background-color example!

This is a text

This is a paragraph

still the div element.

HTML:

```
<h3>CSS background-color example!</h3>
<div>
  This is a text <p>This is a paragraph</p>
  still the div element.
</div>
```

CSS:

```
h3 { background-color:#6495ed; }
div { background-color:#e0ffff; }
p { background-color:#b0c4de; }
```

- Coffee
- Tea
- Coca Cola

HTML:

```
<h3><ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
```

CSS:

```
ul { list-style-image:url('sqpurple.gif'); }
```

Integration von CSS in HTML

Style-Definitionen können an folgenden Stellen definiert werden:

- **Extern**: Definition in einer Datei separat vom HTML-Dokument.
- **Intern**: Definition in einem separaten Bereich innerhalb eines HTML-Dokuments.
- **Inline**: Definition direkt innerhalb eines HTML-Tags.

Cascading: man kann **mehrere Styles gleichzeitig definieren**, welche dann nach einem Regelwerk abgearbeitet und angewendet werden.

Interne Definition – Formatierung für einzelnes Dokument

Definition im <head>-Bereich innerhalb von <style>-Tags.

Beispiel:

```
<head>
  <style>
    h1, h2 {
      color: green;
      font-size: 18px;
    } </style>
  ...
</head>
```

Erklärung: Führt zur Darstellung aller <h1>, <h2> Überschriften in diesem Dokument in der Farbe grün mit einer Grösse von 18 Pixeln.

Externe Definition – Wiederverwendung

- Externe Definitionen können von mehreren HTML-Dateien verwendet werden.
- Einbindung in HTML: `<link rel="stylesheet" href="...">`

Name der CSS-Datei

Beispiel:

mystyle.css:

```
body { background-color: #112233; color: #332211; }  
p { text-transform: capitalize; text-align: center; color: green; }
```

index.html:

```
<head>  
  ..  
  <link rel="stylesheet" href = "mystyle.css" >  
  ..  
</head>  
<body>  
  <p>das ist ein normaler Absatz</p>  
  <p style="color:red;">und der ist noch direkt formatiert</p>  
</body>
```

Inline Styles - Direkte Definition

- "quick-and-dirty"-Formatierung.
- Nur in Notfällen anwenden: Die Übersicht geht schnell verloren, wo man welchen Style direkt angegeben hat.
- Übersteuert alle anderen Formatierungen (interne und externe Styles).

Syntax: `style="prop1: value1; prop2: value2; ..."`

Beispiel:

```
<body style="color: #123456;"  
    ...  
</body>
```

CSS-Selektoren

- Mit einem Style per HTML-Elementtyp lassen sich nur wenige Designs realisieren.
- CSS-Selektoren erlauben eine feingranulare Zuordnung von Styles zu HTML-Elementen.

Selektor	Auswahl	Beispiel
*	jedes Element	<code><p/></code>
element	Element mit dem Namen <i>element</i>	<code><element/></code>
.klasse	Element mit der Klasse <i>klasse</i>	<code><p class="klasse"></code>
#elementid	Element mit der ID <i>elementid</i>	<code><p id="elementid"></code>

ID-Selektor

- Um Elementen individuelle Formate zu geben, verwendet man die individuelle Formatierung.
- Jedes HTML-Element kann ein **eindeutiges** Attribut `id="<name>"` besitzen (Element-ID).
- Die Formatierung erfolgt so (im CSS):

Eindeutige Element-IDs wird man später auch für JavaScript brauchen!

```
#<name> { Eigenschaften; }
```


Beispiel:

CSS:

```
#eins { color:#884422; }
```

HTML:

```
<p id = "eins" >Absatz</p>
```



Klassen-Selektor

- Vereinfachung, falls mehrere Tags denselben Style anwenden sollen.
- Jedes HTML-Element kann ein Attribut `class="<name>"` besitzen.
- Mehrere HTML-Elemente können derselben Klasse angehören.
- Die Formatierung erfolgt so (im CSS):

```
.<name> { Eigenschaften; }
```

Beispiel:

CSS:

```
.absatz {color: red ; }
```

HTML:

```
<h1 class="absatz">...</h1>  
<p class="absatz">...</p>
```


Kombination von Selektoren

- Tag- und Klassen-Selektoren können mittels `<tag>.<klasse>` kombiniert werden (ohne Leerzeichen!).
- **Bedeutung:** Eigenschaften gelten nur `<tag>`-Elemente, welche der `<klasse>` angehören.

Beispiel:

CSS:

```
p.einleitung { font-weight: bold; }  
.einleitung { color: #FF0000; }  
p.absatz { color: #00FF00; }
```

HTML:

```
<p class="einleitung">....</p>  
<p class="absatz">....</p>  
<h1 class="einleitung">..</h1>
```

CSS-Kombinatoren: Wiederverwendbarkeit

Man kann auch für mehrere Tags gleichzeitig ein Style definieren (eine "oder"-Kombination).

Beispiel:

```
...  
h1, h2, h3 { color: blue; }  
...
```

Resultat: Alle Elemente h1, h2 und h3 haben Schriftfarbe blau.

Weitere Kombinatoren (Auswahl)

Kombinator	Auswahl	Beispiel
A B	Element B, wenn es innerhalb von A vorkommt.	<code><A><x></code> <code></x></code> <code></code>
A > B	Element B, wenn es direkt innerhalb von A vorkommt.	<code><A></code> <code></code>
A + B	Element B, wenn es direkt nach A vorkommt (im gleichen Elternelement)	<code><A/></code>
A ~ B	Element B, wenn davor A vorkommt (im gleichen Elternelement)	<code><A/><x/></code> <code></code>

eher selten verwendet


Pseudoklassen-Selektoren

- Definieren Eigenschaften, welche keine Attribute von HTML-Blöcken sind.
- Darstellung von Hyperlinks-Festlegung in CSS

```
a:link      { color:blue; }    /* normaler Link          */
a:visited   { color:green; }   /* bereits besucht        */
a:active     { color:red; }    /* gerade angeklickt      */
a:hover     { color:yellow; }  /* unter dem Mauszeiger   */
a:focus     { color:black; }   /* mit Tastatur angewählt */
```

- Darstellung der ersten Zeile, bzw. des ersten Zeichens:

```
p:first-line { font-weight : bold; }
p:first-letter { font-size : 36pt; color : red;}
```



Man kann nur Brücken schlagen zwischen Ufern die man auseinanderhält. Denn wo es keine Gräben gibt, da gibt es auch keine Unterschiede, und wo es keine Unterschiede gibt, da ist kein Leben.

Attribut-Selektoren (Attribut-bedingte Formatierung)

CSS-Formatierungen lassen sich auf Elemente mit bestimmten Attributen begrenzen.

Beispiele:

1. lege Farbe für alle h1 Überschriften fest, welche ein Alignment gesetzt haben:

```
h1[align] { color:#123456; }
```

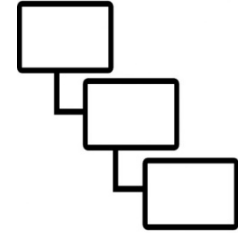
2. stelle alle Absätze, welche ein Namensattribut mit Inhalt "Text" haben, mit Kapitälchen dar:

```
p[name*="Text"] { font-variant:small-caps; }
```

3. weise allen zentriert ausgerichteten HTML-Elementen eine Farbe zu:

```
*[align=center] { color:#654321; }
```

Kaskadierung



- Falls mehrere, sich eventuell widersprechende Styles definiert sind, muss ein Regelwerk zur Anwendung kommen um zu einem Ergebnis zu gelangen.
- Grundprinzip dieses Regelwerkes → **Prioritätsreihenfolge**

(von **hoch** bis **niedrig**)

- | | |
|--------------------|--------------------------------------|
| 1. Gewichtung | (Schlüsselwort !important) |
| 2. Herkunft | (Web-Author, Benutzer, Browser) |
| 3. Besonderheit | (je spezifischer desto mehr Gewicht) |
| 4. Ergebnisabfolge | (Reihenfolge der Regel-Definitionen) |



Prioritätreihenfolge nach Herkunft



1. Deklarationen des Benutzers (*) die !important enthalten.
2. Deklarationen des Web-Authors die !important enthalten.
3. Inline CSS-Anweisungen des Web-Authors.
4. Interne/Externe CSS-Anweisungen des Web-Authors.
5. Deklarationen des Benutzers (*).
6. Deklarationen des Browsers.

(*) Benutzer => Website Besucher → also seine Browsereinstellungen!

Formatierung von Teilbereichen

Was soll man machen, wenn ein (Teil-)Bereich der Seite formatiert werden soll, aber dieser Teil keinem gemeinsamen übergeordneten Tag (z.B. <p>) angehört?

Lösung: Verwendung von Container-HTML-Elementen:

- Es gibt davon sowohl **Inline**- als auch **Block**-Elemente-Varianten.
- Die wichtigsten sind <div> (block) und (inline).

Inline- und Block-Elemente

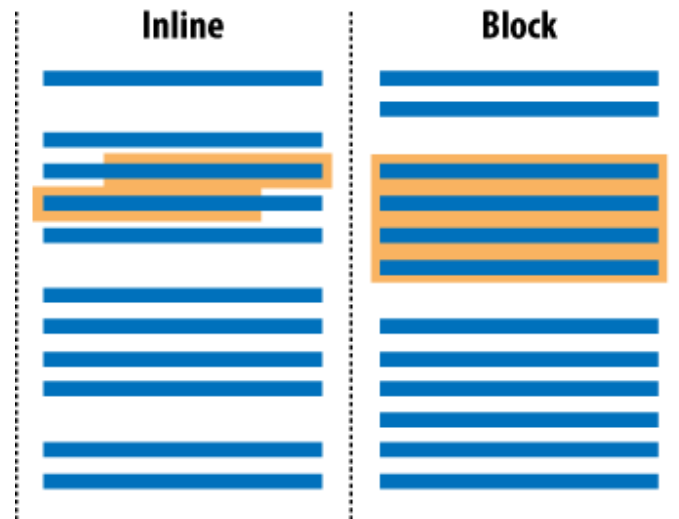
- "Bekannt" aus der Textverarbeitung.

Vereinfacht:

- Inline-Elemente => Zeichenformatierung.
- Block-Elemente => Absatzformatierung.

Ein Block-Element:

- erstreckt sich normalerweise über die gesamte Breite.
- und ist so hoch wie vom Inhalt gefordert (ausser man spezifiziert was anderes).



Inline-Elemente

- Integrieren sich im Textfluss.
- Bevorzugt verwendet um Teilbereiche z.B. eines Textes zu formatieren.
- Verhalten von Elementen kann durch das CSS-Attribut "display" verändert werden.



Beispiel:

```
...Lorem ipsum dolor sit amet, <span style="color:
red;">consectetur adipiscing elit</span>, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua...
```

Block-Elemente

- Führen nach Element einen **Zeilenumbruch** ein.
- Werden bevorzugt zur Gestaltung von Webseiten eingesetzt (Layout).

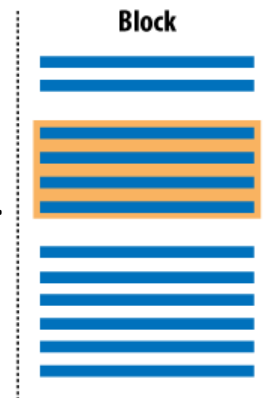
Beispiel:

HTML:

```
<nav id="navigation">  
  <a href="startseite.html">Start</a>  
  <a href="infoseite.html">Info</a>  
  <a href="impressum.html">Impressum</a>  
</nav>
```

CSS:

```
#navigation {  
  position: absolute;  
  top: 100px;  
  left: 100px;  
  color: black;  
}
```

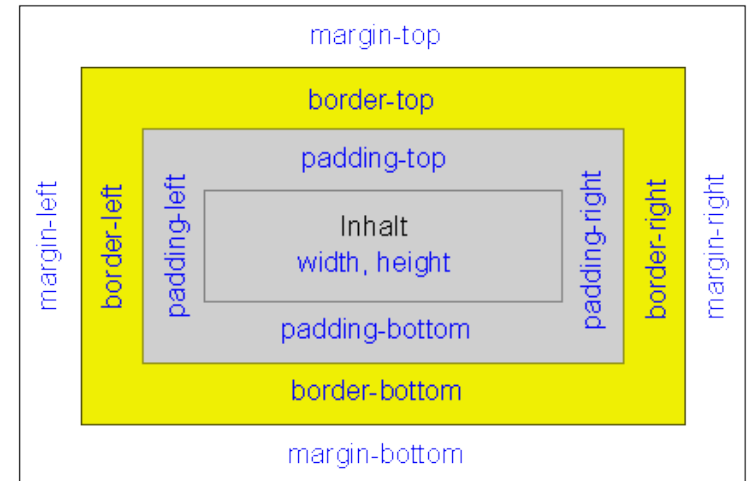


Das Box-Modell

- Beschreibt Innen- und Aussenabstände, Ränder und Inhalt von Block-Elementen
- Abmessungen der Box: Breite und Höhe eines Elements setzt sich **laut der CSS-Regel** wie folgt zusammen:

Gesamte Breite =
linker Aussenabstand + linke Rahmenbreite
+ linker Innenabstand
+ Breite des Inhalts
+ rechter Innenabstand + rechte Rahmenbreite
+ rechter Aussenabstand

(Analog dazu die Höhe)



Box-Modell: Beispiele

margin = Rand (Außenrand/abstand)

- margin: 1px; = allgemein
- margin-left: 1px; = nur links
- margin-right: 1px; = nur rechts
- margin-top: 1px; = nur oben
- margin-bottom: 1px; = nur unten
- margin: 1px 1px; = oben/unten, rechts/links
- margin: 1px 1px 1px 1px; = oben, rechts, unten, links

padding = Polsterung (Innenabstand)

- ähnlich wie oben....

Box-Modell: Beispiele

border-width: 1px; = Rahmendicke

- border-left-width: 1px; etc.

border-style: solid; = Rahmentyp

- | | |
|--------------------------------|----------------|
| - solid | = durchgezogen |
| - dashed | = gestrichelt |
| - dotted | = gepunktet |
| - groove, ridge, inset, outset | = 3D-Effekte |

border-color:red;= Rahmenfarbe

border: 1px solid red; = Kurzform

- border-left: 1px solid red; etc.

Zusammenfassung

- Mit CSS trennt der Entwickler die Präsentation von Inhalt und Struktur.
- CSS basieren auf Regeln:
 - Eine Regel besteht aus einem Selektor und einer Anzahl CSS-Eigenschaften.
 - CSS-Eigenschaften werden auf die selektierten HTML-Elemente angewandt.
 - Kaskadenreihenfolge löst Konflikte zwischen Regeln vorhersehbar auf.
- CSS können extern (typisch), intern und inline definiert werden.
- Wichtige Selektoren: Tag-, ID- und Klasse.
- Kombinationen von Selektoren sind möglich (Stichwort: Kombinatoren).
- Containerelemente zur Definition von Bereichen unterschiedlicher Formatierung.
- Unterscheidung zwischen Block (Seitenumbruch) und Inline (kein Seitenumbruch).
- Styling von Blockelementen mittels Box-Modell (Content, Margin, Border, Padding).