

Web-basierte Systeme

03: Cascading Style Sheets

Wintersemester 2024

Rüdiger Kapitza



Lehrstuhl für Informatik 4
Systemsoftware



Friedrich-Alexander-Universität
Technische Fakultät

HTML & CSS

Motivation von Cascading Style Sheets

In welcher Schriftgröße wird `<h1>Einleitung</h1>` dargestellt?

Motivation von Cascading Style Sheets

In welcher Schriftgröße wird `<h1>Einleitung</h1>` dargestellt?

- In der Standardgröße des Browsers
(HTML legt fest *was* und der Browser definiert das *wie*)

Motivation von Cascading Style Sheets

In welcher Schriftgröße wird `<h1>Einleitung</h1>` dargestellt?

- In der Standardgröße des Browsers
(HTML legt fest *was* und der Browser definiert das *wie*)

Vergangenheit: Standardwerte werden durch Attribute überschrieben

```
1 <table border="2" bordercolor="black">
```

und zusätzliche Tags für die **Darstellung** eingeführt.

```
1 <font face="Arial" color="blue">Einleitung</font>
```

Motivation von Cascading Style Sheets

In welcher Schriftgröße wird `<h1>Einleitung</h1>` dargestellt?

- In der Standardgröße des Browsers
(HTML legt fest *was* und der Browser definiert das *wie*)

Vergangenheit: Standardwerte werden durch Attribute überschrieben

```
1 <table border="2" bordercolor="black">
```

und zusätzliche Tags für die **Darstellung** eingeführt.

```
1 <font face="Arial" color="blue">Einleitung</font>
```

Cascading Style Sheets als flexible Lösung

- Durch CSS wird das *wie* vorgegeben und Standardwerte ersetzt
- Es müssen nicht mehr einzelne Elemente ergänzt werden!

Kernidee: **Separation von Inhalt und Darstellung**

- Darzustellender Inhalt wird in HTML erfasst
- Formatierung wird in separaten CSS Dateien abgelegt
- Vorteile der Aufteilung
 - Einheitliche Gestaltung von mehreren Webseiten
 - Einmal definieren und an einer Vielzahl von Stellen einbinden
 - Ermöglicht Änderungen an einer zentralen Stelle
 - Bspw. neue Corporate Identity erfordert andere Farben
 - Zusätzliche Gestaltungsmöglichkeiten
 - Arbeitsteilung zwischen DesignerInnen (die CSS erstellen) und RedakteurInnen (die Inhalte erstellen)

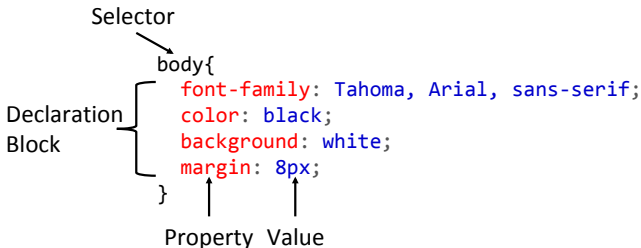
Motivation von Cascading Style Sheets

```
1 <link rel="stylesheet" type="text/css" href="fh.css">
2 </head><body>
3   <h1>Einleitung</h1>
4   <p>Studienziel ist es, ...
5 </body></html>
```

fh.css:

```
1 h1 {
2   font-family: Arial;
3   color: blue;
4 }
5 p:first-letter {
6   font-size: large;
7 }
8
9 body { background-color: green; }
```

Struktur von Stylesheet-Regeln



- Eine Stylesheet-Regel („Rule“) besteht aus einem Selektor welcher eine oder mehrere Deklarationen bündelt
- Der einfachste Selektor besteht aus dem Namen eines einzelnen HTML-Tags

Style für mehrere Seiten

- Einheitliches Aussehen durch das Verwenden der gleichen Stylesheet-Regeln
- Einbindung im head-Bereich einer HTML-Datei

```
1 <link rel="stylesheet" href="mystyle.css">
```

Einbindung spezifisch für eine Seite im Kopf der HTML-Dateien

```
1 <style>
2   h1,h2 {
3     font-family: Arial, Helvetica, sans-serif;
4   }
5 </style>
```

Angaben für einzelne Tags

```
1 <div style="padding:2px; ... ">
```

Gültigkeitsbereich einer Style-Angabe

- Es können innerhalb einer HTML-Datei
 - externe Style Sheet Datei verwendet werden
 - lokale Definition im Kopfbereich durchgeführt werden
 - sowie spezielle Angaben für einzelne Tags erstellt werden
- Widersprechen sich Style-Deklarationen so gilt:
 - Die Angaben bei einem einzelnen Tag haben immer Vorrang
 - Angaben mit mehr spezifischen Selektoren haben Vorrang
 - Sonst gilt die zuletzt angegebene Regel

Möglichkeiten zur Strukturierung & Anwendung von Style-Angaben

- Ziel ist es mehr Flexibilität & keine Einschränkung auf Basistags
- Klassen und IDs zur Gruppierung von Style Angaben
- `<span>` und `<div>` als HTML-Tags zur differenzierten Anwendung von Style-Angaben

Klassen

- In einem Stylesheet können Formatvorlagen – sogenannte “Klassen” – definiert werden

```
1 .wichtig { color: red; }
```

- Name einer Klasse ist frei wählbar und beginnt mit einem Punkt
- HTML-Tags können Klassen zugewiesen werden mittels des Attributs `class`:

```
1 <p>Eine <b class="wichtig">wichtige</b> Meldung</p>  
2 <p class="wichtig">Ein ganz wichtiger Absatz</p>  
3 <p>Ein ganz normaler Absatz</p>
```

- Eine Klasse kann mehrmals in einem Dokument genutzt werden und ein Tag kann **mehrere** Klassen erhalten.

Eindeutige Markierung in einer HTML Datei

- Zur eindeutigen Kennzeichnung von Tags wird das id-Attribut verwendet:

```
1 <p>Eine <b class="wichtig">wichtige</b> Meldung</p>  
2 <p class="wichtig">Ein ganz wichtiger Absatz</p>  
3 <p id="impressum">Das Impressum dieser Seite</p>
```

- Eine ID kann in CSS mittels der Raute referenziert werden:

```
1 #impressum { background-color: #DDD; }
```

Kombinationen aus IDs und Klassen

- Sowohl Klassen (als auch IDs) können mit Tags kombiniert werden um einen komplexen Selektor zu bilden:

```
1 .wichtig { font-size: 20px; }  
2  
3 /* Nur der Tag p mit der Klasse wichtig */  
4 p.wichtig { color: red; }
```

Feingranulare Anwendung von Style-Anweisungen

- Was tun wenn einem Bereich eine bestimmte Klasse zugewiesen werden soll aber kein passender Tag vorhanden ist?
- Hier kann man die beiden Tags `<span>` und `<div>` verwenden, die beide selber kaum Eigenschaft aufweisen.
 - `<span>` eignet sich für die Verwendung in Fließtext
 - `<div>` ist ein blockbildender Tag.

```
1 <p>Es gibt hier ganz <span class="wichtig">besonders  
   interessante</span> Meldungen.</p>
```

Variablen in CSS

- Mit Hilfe von *custom properties* können Werte für CSS-Eigenschaften am Anfang des Stylesheets über Variablen definiert werden.
- Definition unter Verwendung von `--*` und Zugriff via `var( )`

```
1  :root {  
2    --important-color: darkcyan;  
3    --text-on-important-color: darkcyan;  
4  }  
5  
6  h1 {  
7    color: var(--important-color);  
8  }  
9  nav {  
10   background-color: var(--important-color);  
11   color: var(--text-on-important-color);  
12 }
```

Welche Eigenschaften können mittels CSS festgelegt werden?

- Größenangaben
 - Schrift
 - Farben
 - Position
 - Sichtbarkeit
-
- Viele weitere Darstellungsmerkmale, nur ein Teil wird betrachtet!

Relative Maßeinheiten

- `em` Breite des Buchstaben M – also relativ zur Schriftgröße des Elements
- `rem` Entspricht der Schriftgröße, die für das Wurzelement (html-Element) festgelegt wurde
- `vw` Die Einheit vw entspricht dem 100. Teil der Breite des Anzeigebereichs (Viewport)
 - Es gilt also: $100vw$ = Breite des Anzeigebereichs
- `%` Die Einheit % entspricht dem 100. Teil der Breite/Höhe des Elternelements

Absolute Maßeinheiten

- `px` Pixel
- `mm` Millimeter
- `cm` Zentimeter
- `in` Inch

Schrift

```
1  h1,h2 {  
2      font-family: Roboto, sans-serif;  
3      font-size: 18px;  
4      font-weight: bold;  
5      font-style: italic;  
6  
7      letter-spacing: 0.2ex; // Abstand zw. Buchstaben  
8      text-decoration: underline; // unterstrichen  
9      text-shadow: orange 0 -2px; // Schatten Effekt  
10 }  
11 h1 { text-transform: uppercase; } // Großbuchstaben  
12 h2 { font-variant: small-caps; } // Kapitälchen
```

- Schriften müssen entweder im Browser installiert sein oder man verwendet dynamisch zu ladende *Webfonts*

Farben

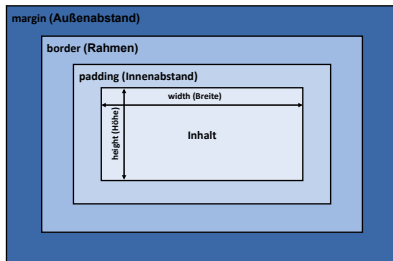
- Farbangaben erfolgen über vordefinierte Farbnamen (red, green,...) oder mit der Angabe von **rot-, grün- und blau-Anteil** in verschiedenen Schreibweisen:
 - Dezimal `rgb(16,0,255)`,
 - zweistellig hexadezimal `#1000FF`.
- Mit CSS3 ist auch die Angabe eines Alpha-Wertes möglich
 - Beispiel: `rgba(153, 134, 117, 0.2)`;
(Braun-Ton, nur zu 20% deckend)

Hintergrund

- Hintergrundfarbe kann mit `background-color` gesetzt werden
- (Anmerkung: `background-image` und `background-repeat` ermöglichen Hintergrundbilder)

Box-Modell

- Jeder blockbildende Tag (z. B. h1, h2, p, blockquote, div, ...) hat einen Rahmen, Innen- und Außenabstand.
- Diese werden mit den Deklarationen border, padding und margin festgelegt.
- Ein Hintergrundbild und/oder eine Hintergrundfarbe des Tags reicht immer bis zum Rahmen, aber nicht darüber hinaus.



Beispiel

```
1 p {  
2   padding: 5px;  
3   margin-top: 5px;  
4   margin-right: 10px;  
5   margin-bottom: 5px;  
6   margin-left: 10px;  
7   border-width: 0px;  
8   border-right-width: 1px;  
9   border-bottom-width: 1px;  
10  background-color: #DDD;  
11 }
```

Alternatives Box-Modell

- Im traditionelle Box Model bezieht sich die Breite (width) auf den **Inhalt**, padding, border und margin muss man erst dazu zählen, um den Gesamt-Platzbedarf zu erhalten.
- Via box-sizing: border-box; kann man auf ein besseres Box-Model umschalten, dann gibt width die Gesamtbreite an.

```
1 box-sizing: border-box;  
2 width: 200px;  
3 padding: 10px;  
4 border-width: 10px;  
5 margin: 32px 0px
```

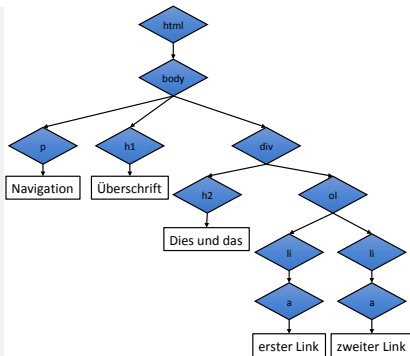
- Gesamtbreite inklusive Rahmen entspricht 200px.
- Platz für den Inhalt: $200\text{px} - (10\text{px} + 10\text{px} + 10\text{px} + 10\text{px}) = 160\text{px}$

- Um Selektoren besser zu verstehen sollte man den *Document Object*-Baum (DOM) genauer betrachten.

```
1 <html>
2 <body>
3 <p>Navigation</p>
4 <h1>Überschrift</h1>
5 <div class="intro">
6 <h2> Dies und das</h2>
7 <ol>
8   <li><a href"...">erster
      Link</a></li>
9   <li><a href"...">zweiter
      Link</a></li>
10 </ol>
11 </div> ...
```

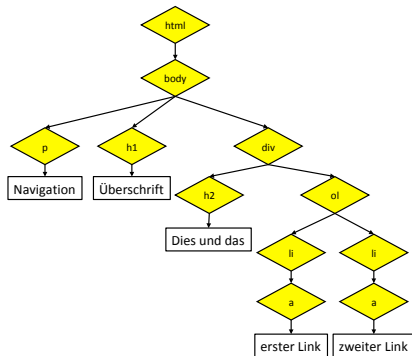
- Um Selektoren besser zu verstehen sollte man den *Document Object*-Baum (DOM) genauer betrachten.

```
1 <html>
2 <body>
3 <p>Navigation</p>
4 <h1>Überschrift</h1>
5 <div class="intro">
6 <h2> Dies und das</h2>
7 <ol>
8   <li><a href"...">erster
      Link</a></li>
9   <li><a href"...">zweiter
      Link</a></li>
10 </ol>
11 </div> ...
```



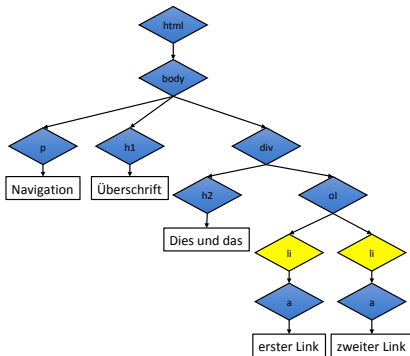
Universal Selector

- Selektor * wählt alle Elemente des Baums aus



Typ Selektor

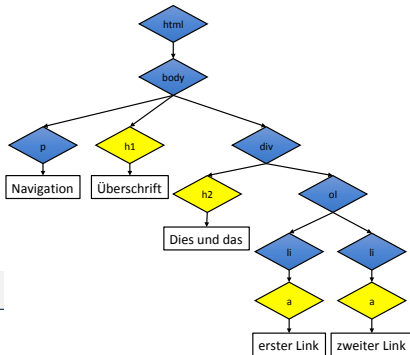
- Über den Namen des HTML-Tags wählt man alle Elemente dieses Typs aus, zum Beispiel wählt `<li>` alle Listen-Elemente aus



Gruppen Selektor

- Mehrere Selektoren können mit Kommas zu einem neuen Selektor gruppiert werden.
- Das Komma entspricht einem “Oder”: selektiert werden Tags die entweder h1 sind, oder h2, oder a:

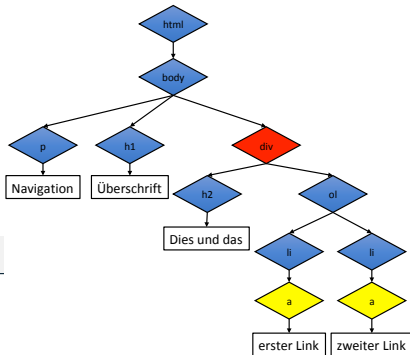
```
1 h1,h2,a { color: red; }
```



Descendant Selektor

- Hier wird ein Element ausgewählt, das Nachkomme eines anderen Elements ist.
- (Achtung: div wird nur zur Auswahl benutzt, wird aber selber nicht ausgewählt!)

```
1  div a { color: red; }
```



Vorrangregel

- Wenn mehrere CSS-Regeln zutreffen wird die Regel mit dem spezifischsten Selektor angewandt
- Typ Selektor wenig spezifisch

```
1 li { color: red; }
```

- Class Selektor mehr spezifisch

```
1 .wichtig { color: red; }
```

- ID Selektor noch mehr spezifisch

```
1 #zuerst { color: red; }
```

- Geerbte Eigenschaften sind minimal spezifisch

```
1 ol { color: red; }
```

Links formatieren

- Man kann die Darstellung von besuchten Links abweichend vom Standardverhalten definieren:

```
1 a:any-link, a:-webkit-any-link { text-decoration:  
    underline; }  
2 a:link { color:blue; }  
3 a:visited { color:#FF00FF; }
```

Interaktion

- Die Pseudo-Klassen `:hover` und `:active` gelten bevor ein Link wirklich geladen wird:
 - `:hover` *schlägt an*, wenn die Maus sich über dem Element befindet
 - Danach wird `:active` wirksam, wenn der Link wirklich ausgelöst wird

Zusammenfassung und Referenzen

Zusammenfassung

- Kernidee: **Separation von Inhalt und Darstellung**
- Nur die '*wichtigsten*' Konzepte von CSS und HTML aufgegriffen
 - CSS ist noch weit detailreicher
 - Entwicklung von Layouts wurde bspw. nicht vorgestellt!

Referenzen

- Web Development (Brigitte Jellinek)
 - <https://web-development.github.io>
- w3schools.com
 - <https://www.w3schools.com/css>
 - Schnelltest
- selfhtml.org – <https://wiki.selfhtml.org>