

## III. Dokumentsprachen

- ❑ Auszeichnungssprachen
- ❑ HTML
- ❑ Cascading Stylesheets CSS
- ❑ XML-Grundlagen
- ❑ XML-Schema
- ❑ Die XSL-Familie
- ❑ APIs für XML-Dokumente

# Cascading Stylesheets CSS

## Einordnung

Ziele von CSS [\[W3C\]](#) :

1. leistungsfähige Layout-Definition für HTML-Dokumente
2. Anpassung an verschiedene Ausgabegeräte/-medien
3. zentrales Layout-Management

Cascading Stylesheets ermöglichen für HTML-Dokumente eine Trennung zwischen Inhalt und Darstellung.

## Bemerkungen:

- CSS kompakt:
  1. Historie
  2. Einbindung von Stylesheet-Information
  3. Stylesheet-Regeln
  4. Selektoren
  5. Deklarationen
  6. Layout
  7. Verarbeitungsstrategie

# Cascading Stylesheets CSS [W3C [status](#), [reports](#), [css home](#)]

## Historie

1996 CSS Level 1. Recommendation.

2011 CSS Level 2 R1. Recommendation.

2016 CSS Level 2 R2. First Public Working Draft. [W3C [WD](#)]

Mit Level 3 beginnt eine modulweise Weiterentwicklung von CSS. [W3C [NOTE](#)]

2012 CSS 3. Aktueller Status der Level 3-Module. [W3C [Status](#)]

2022

2016 CSS 4. Aktueller Status der Level 4-Module. [W3C [Status](#)]

2022

2022 CSS. Aktueller Snapshot. [W3C CSS snapshot [principle](#), [most recent](#)]

## Bemerkungen:

- ❑ Das Wort „Cascading“ bezieht sich auf die kombinierte Anwendung mehrerer Stylesheets. Konflikte zwischen anwendbaren Layout-Vorgaben (d.h. durch die Layout-Vorgaben werden einer Elementinstanz für eine Property für ein Ausgabemedium unverträgliche Werte zugewiesen) werden mit Rücksicht auf Ursprung, Gewichtung und Spezialisierungsgrad sowie Reihenfolge gelöst. [\[W3C\]](#)
- ❑ Die Entwicklung der Cascading Style Sheets geschieht in „Leveln“ (nicht Versionen), die aufeinander aufbauen. Dabei stellen die Features eines höheren Levels eine Übermenge der Features eines niedrigeren Levels dar. Das erlaubte Verhalten für ein Feature in einem höheren Level muss jedoch präziser definiert bzw. weniger tolerant als in einem niedrigeren Level definiert sein. [\[W3C\]](#)
- ❑ CSS Level 2 ist der letzte (gemeinsame) Sprach-Level von CSS. Die Weiterentwicklung nach Level 2 geschieht modulweise. *“There is no CSS Level 4 [and no Level 3].”* [\[W3C NOTE\]](#)
- ❑ Das CSS-Snapshot-Dokument der W3C fasst den State-of-the-Art und damit die offizielle Definition von CSS zusammen. [\[W3C CSS snapshot principle, most recent\]](#)
- ❑ Kurzbeschreibung aller CSS-Spezifikationen: [\[W3C\]](#)
- ❑ CSS Demos. [\[MDN overview, 1, 2, 3\]](#)

# Cascading Stylesheets CSS

Einbindung von Stylesheet-Information [\[MDN\]](#) [\[SELFHTML\]](#)

Die Einbindung bzw. Deklaration von Stylesheet-Information kann auf folgende, miteinander kombinierbare Arten geschehen:

1. Stylesheet-Deklaration in eigener CSS-Datei
2. Stylesheet-Deklaration zentral im HTML-Dokument
3. *Styleattribut*-Deklaration im Start-Tag einer Elementinstanz

# Cascading Stylesheets CSS

Einbindung von Stylesheet-Information [\[MDN\]](#) [\[SELFHTML\]](#)

Die Einbindung bzw. Deklaration von Stylesheet-Information kann auf folgende, miteinander kombinierbare Arten geschehen:

## 1. Stylesheet-Deklaration in eigener CSS-Datei

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

Das `<link>`-Element darf nur im `<head>`-Element verwendet werden.

## 2. Stylesheet-Deklaration zentral im HTML-Dokument

```
<style type="text/css">  
  h3 {color: red; font: arial}  
</style>
```

Das `<style>`-Element darf in dieser Form nur im `<head>`-Element verwendet werden.

## 3. *Styleattribut*-Deklaration im Start-Tag einer Elementinstanz

# Cascading Stylesheets CSS

## Einbindung von Stylesheet-Information [\[MDN\]](#) [\[SELFHTML\]](#)

Die Einbindung bzw. Deklaration von Stylesheet-Information kann auf folgende, miteinander kombinierbare Arten geschehen:

### 1. Stylesheet-Deklaration in eigener CSS-Datei

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

Das `<link>`-Element darf nur im `<head>`-Element verwendet werden.

### 2. Stylesheet-Deklaration zentral im HTML-Dokument

```
<style type="text/css">
  h3 {color: red; font: arial}
</style>
```

Das `<style>`-Element darf in dieser Form nur im `<head>`-Element verwendet werden.

### 3. *Styleattribut*-Deklaration im Start-Tag einer Elementinstanz

```
<h3 style="color: red; font: arial">Neues Kapitel</h3>
```

Die Syntax des **Attributwertes** entspricht dem Deklarationsteil einer CSS-Regel.



## Bemerkungen:

- ❑ Der Geltungsbereich von Stylesheet-Deklarationen, die aus einer CSS-Datei eingebunden werden, ist global für das HTML-Dokument. Die CSS-Datei muss die Namenserweiterung `.css` haben.
- ❑ Der Geltungsbereich von Stylesheet-Deklarationen, die im `<head>`-Element gemacht werden, ist global für das HTML-Dokument.
- ❑ Der Geltungsbereich von *Styleattribut*-Deklarationen ist die Elementinstanz selbst einschließlich ihrer Kindelemente.
- ❑ Im Konfliktfall haben lokale Deklarationen Vorrang vor globalen.
- ❑ Stylesheet-Deklarationen innerhalb eines HTML-Dokuments widersprechen dem Paradigma der Trennung von Inhalt und Darstellung.

## Bemerkungen (Medientypen) :

- Durch Angabe eines `media`-Attributs im `<link>`-Tag lassen sich Medientypen wie `screen`, `print`, `aural`, `braille`, `handheld`, `tv`, `tty` oder `all` spezifizieren. Je nach Endgerät werden vom Browser passende Stylesheets ausgewählt.
- Stylesheet-Dateien lassen sich kombinieren. Beispiel:

```
<link rel="stylesheet" href="base.css">
<link rel="stylesheet" href="print.css" media="print">
<link rel="stylesheet" href="screen1a.css" media="screen">
<link rel="stylesheet" href="screen1b.css" media="screen">
<link rel="alternate stylesheet" href="screen2.css" media="screen" title=...>
```

Im Beispiel gilt das Stylesheet ...

- `base.css` für alle Medientypen.
  - `print.css` zusätzlich für den Medientyp `print`. Die enthaltenen CSS-Regeln werden beispielsweise bei der Erzeugung der Druckvorschau berücksichtigt.
  - `screen1a.css` und `screen1b.css` oder **alternativ** `screen2.css` zusätzlich für den Medientyp `screen`.
- Alternative Stylesheets für den Medientyp `screen` können meist im Browsermenü ausgewählt werden. Als Menüeintrag wird der Text des `title`-Attributs verwendet.



# Cascading Stylesheets CSS

Selektoren [[W3C selectors level 3](#), [level 4](#)] [[MDN](#)]

1. Elementtypselektoren [[W3C](#)]

2. Attributselektoren [[W3C](#)]

3. Klassenselektoren [[W3C](#)]

4. ID-Selektoren [[W3C](#)]

# Cascading Stylesheets CSS

Selektoren [W3C selectors [level 3](#), [level 4](#)] [[MDN](#)]

## 1. Elementtypselektoren [[W3C](#)]

```
h1 {color: red}
```

```
<h1> ... </h1>
```

## 2. Attributselektoren [[W3C](#)]

## 3. Klassenselektoren [[W3C](#)]

## 4. ID-Selektoren [[W3C](#)]

# Cascading Stylesheets CSS

Selektoren [W3C selectors [level 3](#), [level 4](#)] [[MDN](#)]

## 1. Elementtypselektoren [[W3C](#)]

```
h1 {color: red}
```

```
<h1> ... </h1>
```

## 2. Attributselektoren [[W3C](#)]

```
[hreflang=fr] {color: red}
```

```
<a hreflang="fr"> ... </a>
```

```
h1[lang] {color: red}
```

```
<h1 lang="fr"> ... </h1>
```

## 3. Klassenselektoren [[W3C](#)]

## 4. ID-Selektoren [[W3C](#)]

# Cascading Stylesheets CSS

Selektoren [W3C selectors [level 3](#), [level 4](#)] [[MDN](#)]

## 1. Elementtypselektoren [[W3C](#)]

```
h1 {color: red}
```

```
<h1> ... </h1>
```

## 2. Attributselektoren [[W3C](#)]

```
[hreflang=fr] {color: red}
```

```
<a hreflang="fr"> ... </a>
```

```
h1[lang] {color: red}
```

```
<h1 lang="fr"> ... </h1>
```

## 3. Klassenselektoren [[W3C](#)]

```
.Classname {color: red}
```

```
<div class="Classname"> ... </div>
```

```
h1.Classname {color: red}
```

```
<h1 class="Classname"> ... </h1>
```

## 4. ID-Selektoren [[W3C](#)]

# Cascading Stylesheets CSS

Selektoren [W3C selectors [level 3](#), [level 4](#)] [[MDN](#)]

## 1. Elementtypselektoren [[W3C](#)]

```
h1 {color: red}
```

```
<h1> ... </h1>
```

## 2. Attributselektoren [[W3C](#)]

```
[hreflang=fr] {color: red}
```

```
<a hreflang="fr"> ... </a>
```

```
h1[lang] {color: red}
```

```
<h1 lang="fr"> ... </h1>
```

## 3. Klassenselektoren [[W3C](#)]

```
.Classname {color: red}
```

```
<div class="Classname"> ... </div>
```

```
h1.Classname {color: red}
```

```
<h1 class="Classname"> ... </h1>
```

## 4. ID-Selektoren [[W3C](#)]

```
#Identifier {color: red}
```

```
<div id="Identifier"> ... </div>
```

```
h1#Identifier {color: red}
```

```
<h1 id="Identifier"> ... </h1>
```



# Cascading Stylesheets CSS

## Selektoren (Fortsetzung)

5. Pseudo-Klassenselektoren [[W3C](#)]

6. Pseudo-Elementselektoren [[W3C level 3](#), [level 4](#)]

7. Kombinierte Selektoren [[W3C](#)]

# Cascading Stylesheets CSS

## Selektoren (Fortsetzung)

### 5. Pseudo-Klassenselektoren [\[W3C\]](#)

```
:hover {color: red}
```

```
<h1> ... </h1>
```

```
a:visited {color: red}
```

```
<a href="./sub/page1.html"> ... </a>
```

### 6. Pseudo-Elementselektoren [\[W3C level 3, level 4\]](#)

### 7. Kombinierte Selektoren [\[W3C\]](#)

# Cascading Stylesheets CSS

## Selektoren (Fortsetzung)

### 5. Pseudo-Klassenselektoren [\[W3C\]](#)

```
:hover {color: red}
```

```
<h1> ... </h1>
```

```
a:visited {color: red}
```

```
<a href="./sub/page1.html"> ... </a>
```

### 6. Pseudo-Elementselektoren [\[W3C level 3, level 4\]](#)

```
::first-letter {color: red}
```

```
<p> ... </p>
```

```
p::first-line {color: red}
```

```
<p> ... </p>
```

### 7. Kombinierte Selektoren [\[W3C\]](#)

# Cascading Stylesheets CSS

## Selektoren (Fortsetzung)

### 5. Pseudo-Klassenselektoren [\[W3C\]](#)

```
:hover {color: red}
```

```
<h1> ... </h1>
```

```
a:visited {color: red}
```

```
<a href="./sub/page1.html"> ... </a>
```

### 6. Pseudo-Elementselektoren [\[W3C level 3, level 4\]](#)

```
::first-letter {color: red}
```

```
<p> ... </p>
```

```
p::first-line {color: red}
```

```
<p> ... </p>
```

### 7. Kombinierte Selektoren [\[W3C\]](#)

```
h1 em {color: red}
```

```
<h1>This is <em>very</em> ... </h1>
```

```
thead > tr {color: orange}
```

```
<thead> <tr> ... </thead> [Beispiel]
```

```
em + em {color: red}
```

```
<h1><em>This</em> and <em>that</em> </h1>
```

```
h1 ~ p {color: orange}
```

```
<h1>... </h1> ... <p> ... </p>
```

## Bemerkungen:

- ❑ Pseudoklassen ermöglichen die Selektion auf Basis von Information, die nicht oder nur versteckt im Dokumentenbaum abgebildet ist. [\[W3C\]](#)

Diese Information kann sich auf folgende Aspekte beziehen:

- Links und Lokationen [\[W3C\]](#)
  - Benutzeraktionen [\[W3C\]](#)
  - Zeitverläufe [\[W3C\]](#)
  - linguistische Konzepte [\[W3C\]](#)
  - Eingabezustände [\[W3C\]](#)
  - Dokumentbaumstruktur [\[W3C\]](#)
- ❑ Pseudoelemente ermöglichen die Selektion von Dokumenteninhalten, die nicht durch die Markup-Sprache ausgezeichnet werden kann. [\[W3C\]](#)  
Beispiele: der erste Buchstabe eines Abschnitts, die letzte Zeile eines Abschnitts.
  - ❑ Unter die Rubrik der kombinierten Selektoren fällt insbesondere die Spezifikation von Nachbarschaftsbeziehungen im Dokumentenbaum. [\[W3C\]](#)  
Beispiele:  $\sqsubset$  (ist Nachfolger von),  $\succ$  (ist Kindknoten von),  $\sim$  (ist Geschwisterknoten von).

# Cascading Stylesheets CSS

## Selektoren (Fortsetzung)

Mit Hilfe von Klassenselektoren kann die Einführung neuer Elementtypen nachempfunden werden.

Besonders geeignet sind die **funktionslosen** HTML-Elementtypen [\[MDN\]](#):

1. Block-Elementtyp `<div>`
2. Inline-Elementtyp `<span>`

# Cascading Stylesheets CSS

## Selektoren (Fortsetzung)

Mit Hilfe von Klassenselektoren kann die Einführung neuer Elementtypen nachempfunden werden.

Besonders geeignet sind die **funktionslosen** HTML-Elementtypen [\[MDN\]](#):

1. Block-Elementtyp `<div>`
2. Inline-Elementtyp `<span>`

Beispiel:

```
<head>
  <title>Example</title>
  <style type="text/css">
    .my-h1 {font-family: sans-serif; color: blue}
  </style>
</head>
<body>
  <div class="my-h1">Ein eigener Stil für h1-Überschriften</div>
  ...
```

## Bemerkungen:

- ❑ HTML verwendet eine feste Dokumentstruktur und somit sind alle Elementtypen vorgegeben. Diese HTML-Elementtypen besitzen elementtypspezifische Vorgaben für ihre Darstellung.
- ❑ Die Schaffung von Elementinstanzen, die zu einer gemeinsamen Stylesheet-Klasse gehören und darüberhinaus ohne weitere Funktion (= ohne intendierte Semantik) sind, geschieht in zwei Schritten:
  1. Definition einer neuen Stylesheet-Klasse mittels `.Classname { ... }`.
  2. Verwendung der neuen Stylesheet-Klasse mittels `class="Classname"` in Elementinstanzen des Typs `<div>` oder `<span>`.
- ❑ Das W3C warnt vor der übertriebenen Nutzung dieser Möglichkeit, weil die intendierte Semantik selbstdefinierter Klassen für Außenstehende oft nicht erkennbar ist. [\[W3C\]](#)

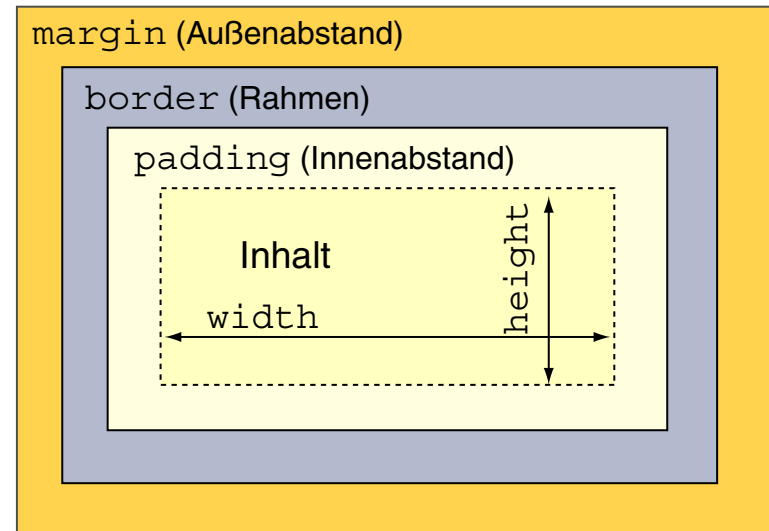


# Cascading Stylesheets CSS

## Deklarationen

CSS-Deklarationen können sich auf nahezu alle Aspekte der Dokumentgestaltung beziehen. [[SELFHTML](#)]

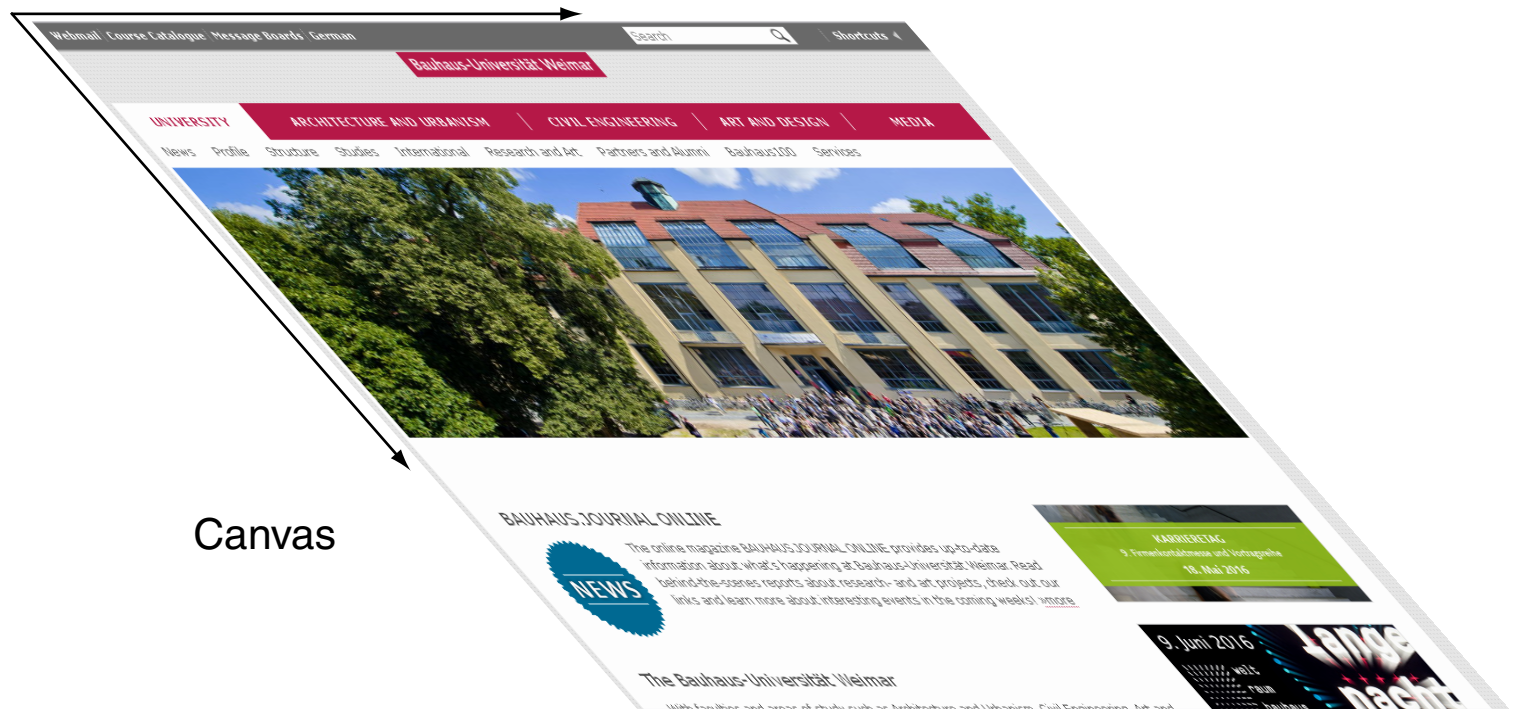
- ❑ Maßeinheiten [[MDN](#)]
- ❑ Schrift [[MDN](#)]
- ❑ Farben [[MDN](#)]
- ❑ Listen [[MDN](#)]
- ❑ Tabellen [[MDN](#)] [[webis aitoools](#), [lecturenotes](#)]
- ❑ Box-Modell [[W3C](#)] [[MDN](#)] [[SELFHTML](#)]



# Cascading Stylesheets CSS

Layout (*Visual Formatting*) [W3C [CSS 2.2](#), [flexbox](#)]

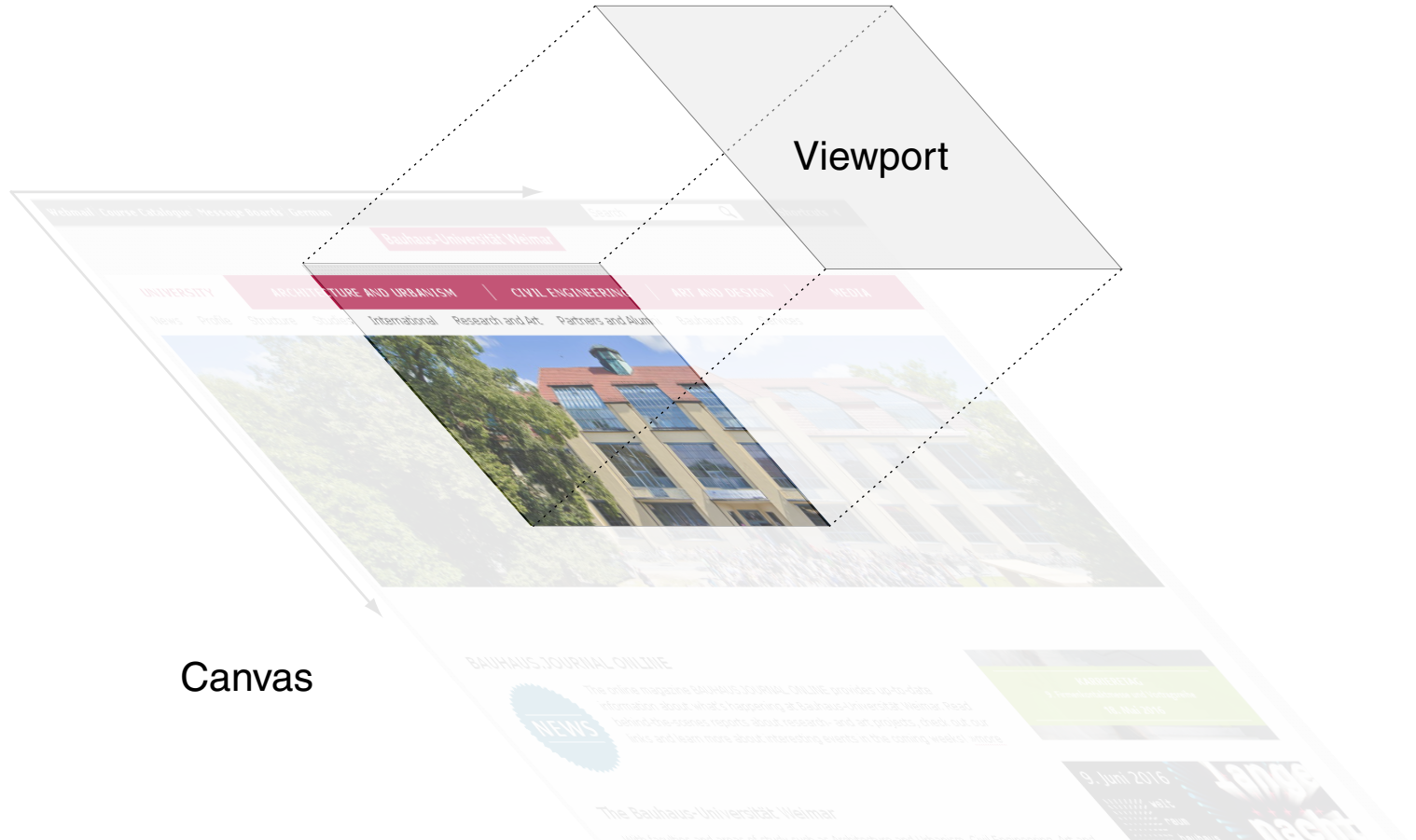
Das Rendering eines Dokuments erfolgt auf einer unendlich großen Zeichenfläche (*Canvas*). Der Viewport zeigt einen Ausschnitt der Zeichenfläche.



# Cascading Stylesheets CSS

Layout (*Visual Formatting*) [W3C [CSS 2.2](#), [flexbox](#)]

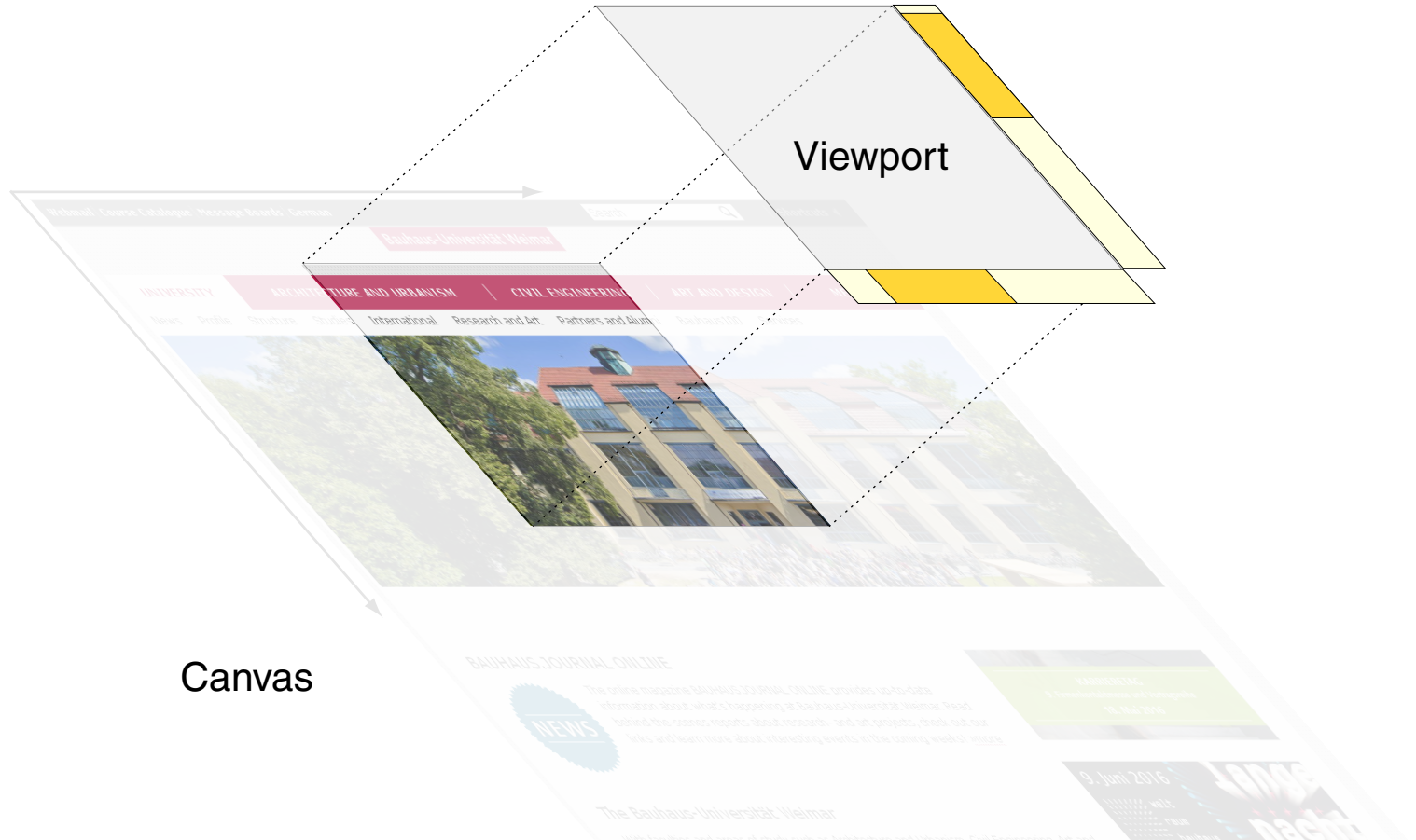
Das Rendering eines Dokuments erfolgt auf einer unendlich großen Zeichenfläche (*Canvas*). Der Viewport zeigt einen Ausschnitt der Zeichenfläche.



# Cascading Stylesheets CSS

Layout (*Visual Formatting*) [W3C [CSS 2.2](#), [flexbox](#)]

Das Rendering eines Dokuments erfolgt auf einer unendlich großen Zeichenfläche (*Canvas*). Der Viewport zeigt einen Ausschnitt der Zeichenfläche.



# Cascading Stylesheets CSS

## Layout (*Visual Formatting*) (Fortsetzung)

### □ `display`-Property [\[W3C\]](#)

---

<code>block</code>	Element erzeugt eine Block-Box.
<code>inline</code>	Element erzeugt eine oder mehrere Inline-Boxen.
<code>inline-block</code>	Element erzeugt eine nicht teilbare Inline-Box, der Inhalt wird im Block-Kontext formatiert.

---

### □ `position`-Property [\[W3C\]](#)

---

<code>static</code>	Box wird im vorliegenden Kontext (Block/Inline) gesetzt.
<code>relative</code>	Box wird mit Offset relativ zur normalen Position gesetzt.
<code>absolute</code>	Box wird an absoluter Position im enthaltenen Block gesetzt.
<code>fixed</code>	Box wird an absoluter Position im enthaltenen Block gesetzt, aber in port-Ebene. (Position fest im Viewport)

---

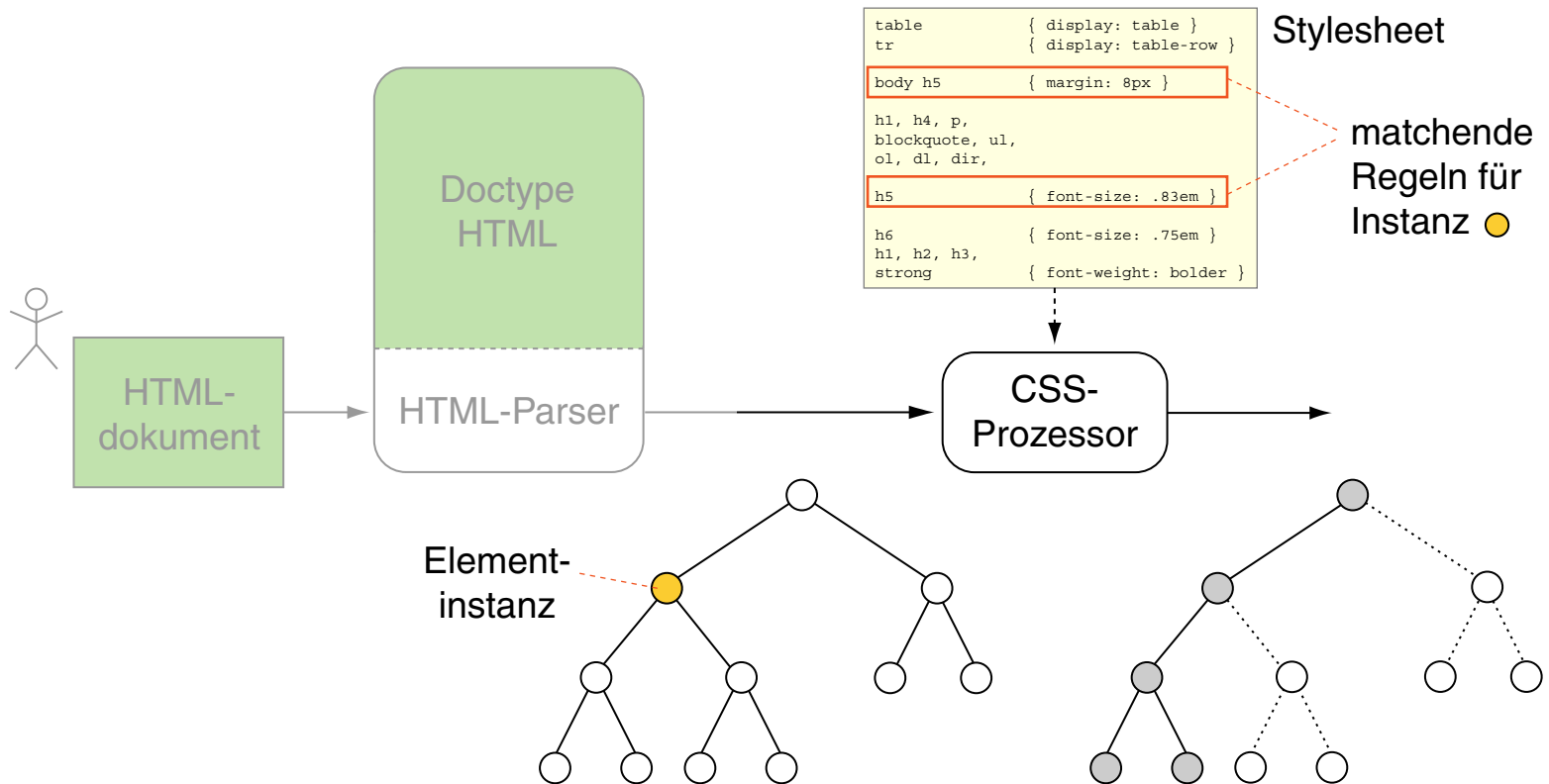
### □ **Offsets** `top`, `right`, `bottom`, `left` [\[W3C\]](#)

Angabe der Position als Abstand vom jeweiligen Rand der enthaltenden Box (`absolute/fixed`) oder von der vorgesehenen Position der Box (`relative`).

# Cascading Stylesheets CSS

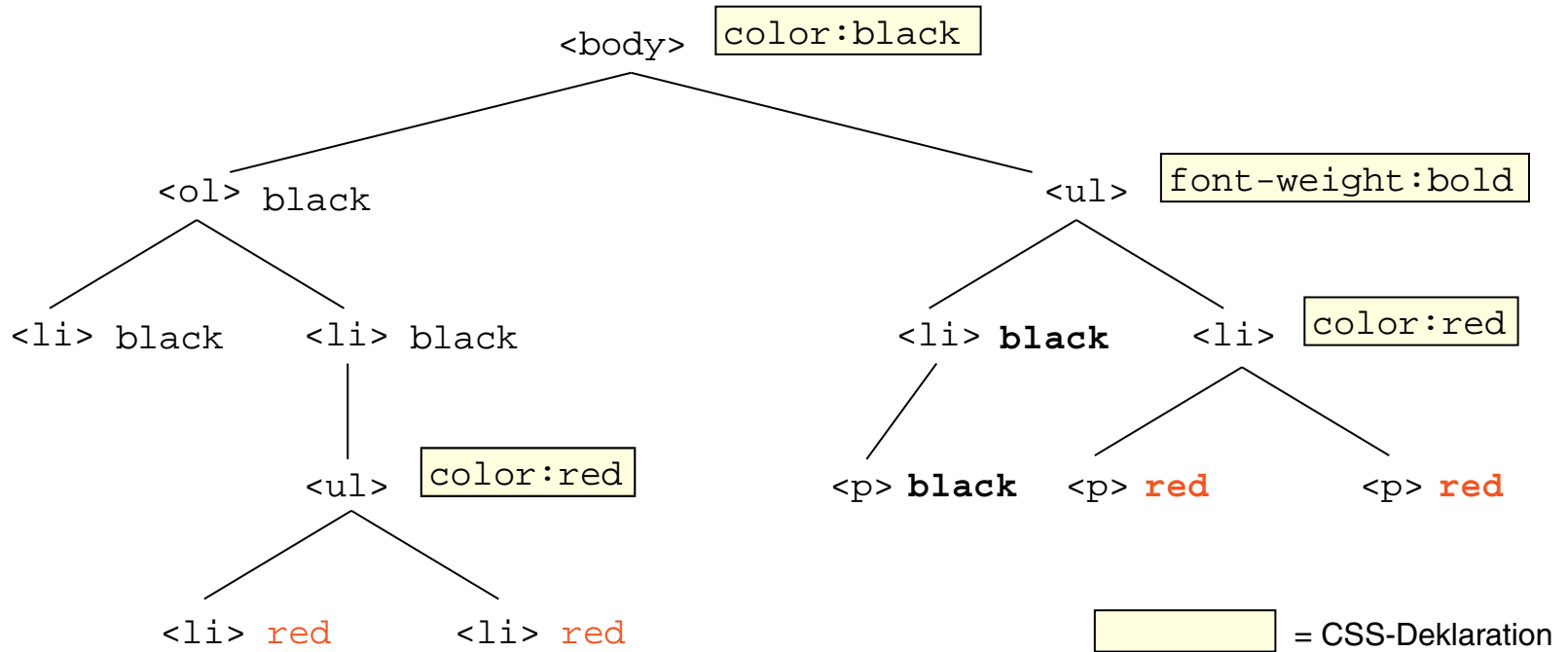
Verarbeitungsstrategie [\[MDN\]](#) [\[WT:III XSL-Verarbeitung\]](#)

Der Dokumentbaum (DOM) wird in Pre-Order-Reihenfolge verarbeitet. Zu jeder Elementinstanz werden alle matchenden CSS-Regeln in den Stylesheets identifiziert, gemäß des Cascade-Algorithmus sortiert und angewandt.



# Cascading Stylesheets CSS

## Verarbeitungsstrategie (Fortsetzung)



□ Deklarationen werden an eingebettete Elementinstanzen vererbt.

□ Lokale Vorgaben überschreiben vererbte Werte und Defaults:

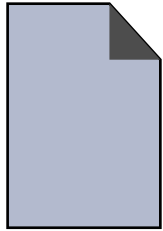
`color:black` → `red`

`font-weight:normal` → `bold`

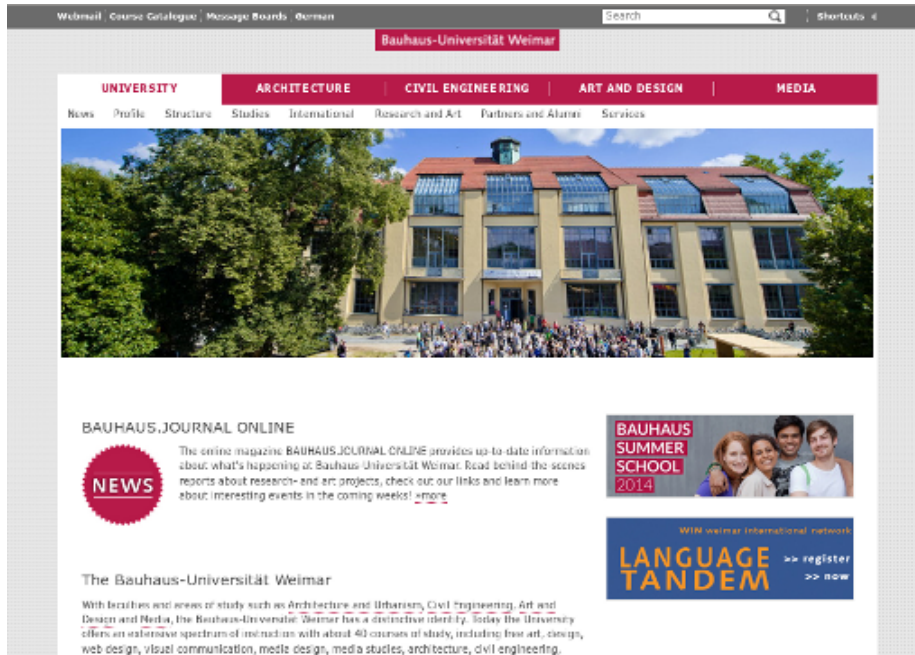
# Cascading Stylesheets CSS

Verarbeitungsstrategie (Fortsetzung) [W3C] [SELFHTML]

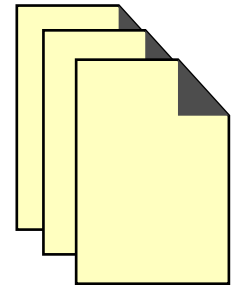
Für die Darstellung von HTML-Dokumenten werden drei Arten von Stylesheets ausgewertet:



Browser-Stylesheet



Benutzer-Stylesheet



Autoren-Stylesheets



# Cascading Stylesheets CSS

## Verarbeitungsstrategie (Fortsetzung)

Für die Darstellung von HTML-Dokumenten werden drei Arten von Stylesheets ausgewertet:

### 1. Browser-Stylesheet

Definiert das Standard-Layout für die Elementinstanzen; dieses Stylesheet ist Browser-spezifisch und wird vom Browser-Hersteller entwickelt. Ein entsprechender W3C-Vorschlag befindet hier [\[W3C\]](#).

### 2. Benutzer-Stylesheet

Definiert die Präferenzen eines Benutzers. Die Spezifikation des Benutzer-Stylesheets geschieht über einen Browser-Dialog.

### 3. Autoren-Stylesheet(s)

Die „eigentlichen“ (sichtbaren) Stylesheets, die ein Autor eines HTML-Dokuments zur Realisierung seiner Layout-Ziele entwickelt hat.

## Bemerkungen:

- Regeln lassen sich durch Angabe von `!important` stärker gewichten. Im Beispiel ist das Setzen der Property `font-style` stärker gewichtet:

```
p {  
  font-style: italic !important;  
  color: red;  
}
```

- Eine Gewichtung ist nur für Autoren- und Benutzer-Stylesheets spezifizierbar.
- Konflikte zwischen anwendbaren Layout-Vorgaben werden zunächst mit Rücksicht auf
  1. Ursprung und Gewichtung gelöst [\[W3C\]](#) :
    - (a) `!important`-Regeln aus Benutzer-Stylesheet vor
    - (b) `!important`-Regeln aus Autoren-Stylesheets vor
    - (c) normale Regeln aus Autoren-Stylesheets vor
    - (d) normale Regeln aus Benutzer-Stylesheet vor
    - (e) Regeln aus Browser-Stylesheet.

Bestehen immer noch Konflikte, so werden diese mit Rücksicht auf

2. Spezialisierungsgrad (spezifischere Regeln vor allgemeineren, s.u.) und
3. Reihenfolge (spätere Regeln vor früheren) gelöst.

Nur eine Layout-Vorgabe wird umgesetzt.

- Algorithmus zur Bestimmung der Regelspezifität: [\[W3C\]](#) [\[calculator\]](#)

# Cascading Stylesheets CSS

## Quellen zum Nachlernen und Nachschlagen im Web

- ❑ MDN. *CSS*.  
[developer.mozilla.org/en-US/docs/Web/CSS](https://developer.mozilla.org/en-US/docs/Web/CSS)
- ❑ MDN. *CSS Tutorial*.  
[developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting\\_started](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started)
- ❑ W3C. *CSS Level 2 R2*.  
[www.w3.org/TR/CSS2](http://www.w3.org/TR/CSS2)
- ❑ W3C. *CSS Home*.  
[www.w3.org/Style/CSS](http://www.w3.org/Style/CSS)
- ❑ W3C. *CSS Validation Service*.  
[jigsaw.w3.org/css-validator](http://jigsaw.w3.org/css-validator)
- ❑ W3 Schools. *CSS*.  
[www.w3schools.com/css](http://www.w3schools.com/css)