CSS - Cascading Style Sheets

Agenda

Grundlagen

Selektoren

Einheiten

Verschiedene CSS Eigenschaften

Pseudo-Klassen und -Elemente

Layout-Systeme

Assignment

Grundlagen

Einführung in CSS

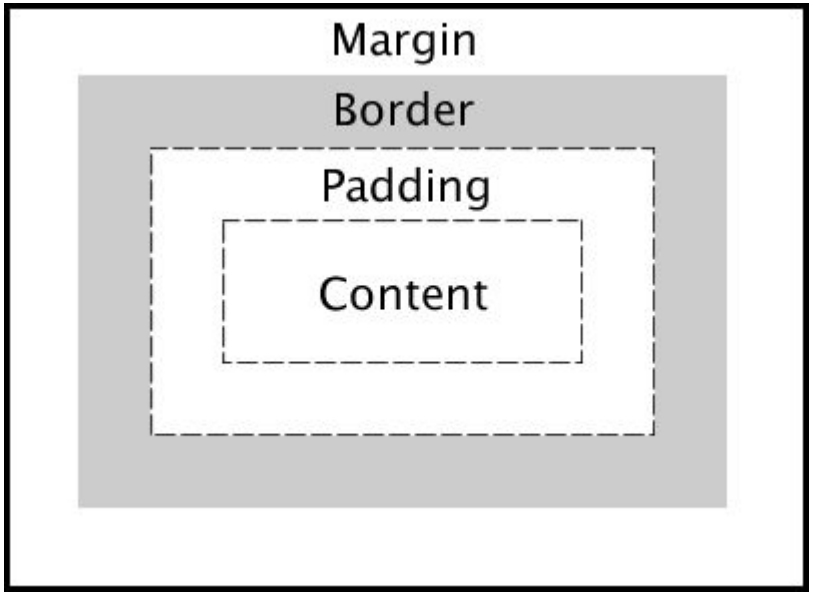
- Definition: CSS (Cascading Style Sheets) ist eine Stylesheet-Sprache, die für die Beschreibung des Aussehens und der Formatierung von Dokumenten verwendet wird, die in einer Markup-Sprache wie HTML geschrieben sind.
- Zweck: CSS ermöglicht es Entwicklern und Designern, das Design und Layout von Webseiten zu kontrollieren.
- CSS trennt den Inhalt (HTML) vom Design, was die Wartung und das Webdesign vereinfacht.
- Wird vom W3C entwickelt
- Browser muss CSS unterstützen → Unterschiede bei verschiedenen Browsern / Versionen

Syntax

- Syntax für ein Regelset: Selektor { Eigenschaft: Wert; }
- Selektoren: Der Teil eines CSS-Rulesets, der bestimmt, auf welche Elemente der Stil angewendet wird
- Deklarationen: Ein Style-Regel, bestehend aus einem Eigenschaftsnamen und einem Wert (z.B. “color: red;”)
- Regelset: Eine Kombination aus Selektor(en) und einer Deklaration
- Mehrere Selektoren und mehrere Deklarationen in einem Regelset möglich

Box-Modell

- Jedes HTML Element als Box zu betrachten
- CSS Properties for the Box:
  - margin
  - padding
  - border



CSS in HTML einbinden

Externe Stylesheets: Mittels <link> Element im <head>-Bereich.

```
<head>
    <title>Einbindung von CSS </title>
    <link rel="stylesheet" href="style.css" >
</head>
```

Interne Stylesheets: Mit <style>-Tags im <head>-Bereich.

```
<head>
    <title>Einbindung von CSS </title>
    <style type="text/css" >
        h1 { color: red; }
    </style>
</head>
```

Inline-Stile: Direkt im style-Attribut von HTML-Elementen.

```
<p style="color: red;">This is red</p>
```

Selektoren

Selektoren

Selektor	Beispiel
Universal-Selektor	`* { color: purple; }` `/* alle Elemente */`
Element-Selektor	`h1 { color: red; }` `/* alle h1 Elemente */`
Klassen-Selektor	`.some-class { color: blue; }` `/* alle Elemente mit der Klasse some-class */` `h1.some-class { color: blue; }` `/* alle h1 Elemente mit der Klasse some-class */`
ID-Selektor	`#some-id { color: green; }` `/* das Element mit der ID some-id */` `h1#some-id { color: green; }` `/* das h1 Element mit der ID some-id */`
Nachfahren-Selektor	`h1 i { color: yellow; }` `/* alle i Elemente, die in einem h1 Element sind (rekursiv) */` `h1 * i { color: yellow; }` `/* alle i Elemente, die in direkten Unterelementen eines h1 Elements sind */`

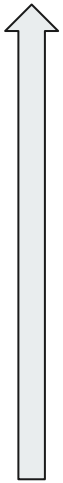
Selektoren

Selektor	Beispiel
Kind-Selektor	`h1 > i { color: yellow; } /* alle i Elemente, die direkte Unterelemente eines h1 Elements sind */`
Nachbar-Selektor	`h1 + p { color: yellow; } /* alle p Elemente, die direkt nach einem h1 Element folgen */`

Attribut-Selektor	`h1[align] { color: yellow; } /* alle h1 Elemente mit einem align Attribut */`
	`h1[align=center] { color: yellow; } /* alle h1 Elemente mit einem align Attribut und dem Wert center */`
	`h1[align|=center] { color: yellow; } /* alle h1 Elemente mit einem align Attribut dessen Wert mit center beginnt... */`
	`h1[align~=center] { color: yellow; } /* alle h1 Elemente mit einem align Attribut dessen Wert center enthält */`

Mehrere Selektoren	`h1, .some-class, h1 i { color: yellow; } /* die Ergebnisse aller Selektoren */`

Priorität der Selektoren



Definition in HTML style Attribut

ID-Selektor

Klassen-Selektor

Element-Selektor

Übung 1

Erstelle ein HTML Dokument mit dem Grundgerüst Der HTML Body soll folgende Elemente enthalten:

- 1 <p> Element mit der Klasse “loremipsum” und einem Text
- 1 <p> Element ohne Klasse und einem anderen Text
- 1 Element mit der ID “dolorsitamet” und einem Text
- 1 Element mit der Klasse “loremipsum” und einem Text

Erstelle ein separates CSS Dokument im gleichen Ordner und lade es in der HTML Datei, um die Styles dort zu benutzen. Folgende Regeln sollen implementiert werden:

- Der Text aller <p> Elemente soll blau sein.
- Der Text aller <p> Elemente mit der Klasse “loremipsum” soll rot sein.
- Der Text des Elements mit der ID “dolorsitamet” soll grün sein.

Einheiten

Absolute Einheiten in CSS

pt	Punkt (1/72 Inches)	`p { font-size: 10pt; } /* 10pt means 1/72 * 10 = 0,138 inches * 2,54 = 0,35cm */`
pc	Pica (12 Punkte)	`p { font-size: 10pc; } /* 10pc means 10 * 12 = 120pt */`
in	Inch (2,54 cm)	`p { font-size: 1in; } /* 1in means 1 * 2,54 = 2,54cm */`
mm	Millimeter	`p { font-size: 1mm; }`
cm	Zentimeter	`p { font-size: 1cm; }`

Relative Einheiten in CSS

Empfehlung vom W3C für Bildschirme: em, %, px, vh, vw

px	Pixel	`div { width: 10px; } /* 10px means 10 pixels */`
em	Relativ zur Schriftgröße des Elements	`div { width: 2em; } /* 2em means 2 times of font size */`
ex	Relativ zur Höhe von Kleinbuchstaben des Elements	`div { width: 2ex; } /* 2ex means 2 times of x-height */`
rem	Relativ zur Schriftgröße des Root-Elements	`div { width: 2rem; } /* 2rem means 2 times of root element font size */`
%	Relativ zum Elternelement	`div { width: 10%; } /* 10% means 10% of parent element */`
vh	Relativ zur Höhe des Browserfensters	`div { width: 10vh; } /* 10vh means 10% of viewport height */`
vw	Relativ zur Breite des Browserfensters	`div { width: 10vw; } /* 10vw means 10% of viewport width */`

Farben

Farben in CSS können unterschiedliche spezifiziert werden:

Namen	`h1 { color: red }`
RGB	`h1 { color: rgb(255, 99, 71) }`
RGBA	`h1 { color: rgba(255, 99, 71, 0.6) }`
HSL	`h1 { color: hsl(9, 100%, 64%) }`
HSLA	`h1 { color: hsla(9, 100%, 64%, 0.6) }`
HEX	`h1 { color: #ff6347 }`

Verschiedene CSS Eigenschaften

Textformatierung Eigenschaften

font-family	Schriftart(en)	`p { font-family:'Segoe UI', Tahoma, sans-serif; }`
font-style	italic / oblique / normal	`p { font-style: italic; }`
font-variant	small-caps / normal	`p { font-variant: small-caps; }`
font-weight	bold / bolder / lighter / normal / 1-9 * 100	`p { font-weight: bold; }`
font-size	Schriftgröße	`p { font-size: 12pt; }`
color	Schriftfarbe	`p { color: blue; }`

Textformatierung Eigenschaften

font	Kombinierte Angabe	`p { font: italic small-caps bold 12pt/14pt 'Segoe UI', Tahoma, sans-serif; }`
word-spacing	Wortabstand	`p { word-spacing: 1px; }`
letter-spacing	Zeichenabstand	`p { letter-spacing: 1px; }`
text-decoration	underline / overline / line-through / blink / none	`p { text-decoration: underline; }`
text-transform	capitalize / uppercase / lowercase / none	`p { text-transform: uppercase; }`

Ausrichtung und Abstände

text-indent	Einrückung	`div { text-indent: 5px; }`
line-height	Zeilenhöhe	`div { line-height: 5px; }`
vertical-align	top / middle / bottom / baseline / sub / super / text-top / text-bottom	`div { vertical-align: middle; }`
text-align	left / center / right / justify	`div { text-align: justify; }`
white-space	normal (default): automatischer Umbruch nowrap: kein automatischer Zeilenumbruch pre: Zeilenumbruch wie im Editor pre-wrap: wie ^ + Umbruch am Boxrand pre-line: wie ^ + Zf. von White Spaces	`div { white-space: nowrap; }`

Ausrichtung und Abstände

Richtungen: top, right, bottom, left

margin-[Richtung]	Außenabstand	`div { margin-top: 10px; }` `div { margin-top: inherit; } /* inherit from parent element */`

margin	Allgemeine Angabe	`div { margin: 10px; } /* 10px top, right, bottom, left */`
		`div { margin: 10px 20px; } /* 10px top + bottom, 20px left+right */`
		`div { margin: 10px 20px 30px; } /* 10px top, 20px left+right, 30px bottom */`
		`div { margin: 10px 20px 30px 40px; } /* 10px top, 20px right, 30px bottom, 40px left */`

padding-[Richtung]	Innenabstand	`div { padding-top: 10px; }` `div { padding-top: inherit; } /* inherit from parent element */`
padding	Allgemeine Angabe	Siehe margin

Rahmen

border-[Richtung]-width	`div { border-width: 10px 5px 10px 5px; } /* top, right, bottom, left */` `div { border-top-width: 10px; }`
border-[Richtung]-color	`div { border-color: blue red green brown; } /* top, right, bottom, left */` `div { border-top-color: blue; }`
border-[Richtung]-style	`div { border-style: solid dashed dotted double; } /* top, right, bottom, left */` `div { border-top-style: solid; }`
border-[Richtung]	`div { border: 10px solid blue; } /* width, style, color */` `div { border-top: 10px solid blue; }`

Hintergrund

background-color	`div { background-color: red; }`
background-image	`div { background-image: url("image.jpg"); }`
background-repeat	`div { background-repeat: no-repeat; } /* repeat (default), repeat-x, repeat-y */`
background-attachment	`div { background-attachment: fixed; } /* scroll (default), fixed */`
background-position	`div { background-position: 10px 20px; } /* x, y (keywords: top, bottom, center, left, right) */`
background	`div { background: red url("image.jpg") no-repeat fixed 10px 20px; }`

Tabellen

border-collapse	`table { border-collapse: separate; } /* or collapse (default) */`
border-spacing	`table { border-spacing: 5px; }`
empty-cells	`table { empty-cells: show; } /* or hide (default) */`
caption-side	`table { caption-side: bottom; } /* or top (default) */`
table-layout	`table { table-layout: fixed; } /* or auto (default) */`

Listen

list-style-type	`ol { list-style-type: disc; } /* decimal, lower-roman, upper-roman, lower-alpha, upper-alpha */` `ul { list-style-type: square; } /* circle, disc, none */`
list-style-position	`ol { list-style-position: inside; } /* outside (default) */`
list-style-image	`ul { list-style-image: url("image.jpg"); }`

Positionierung

top	Verschiebt Element von oben	`div { top: 10px; }`
left	Verschiebt Element von links	`div { left: 10px; }`
bottom	Verschiebt Element von unten	`div { bottom: 10px; }`
right	Verschiebt Element von rechts	`div { right: 10px; }`

Positionierung

position	**static**: Positionierung hat keinen Effekt → Element nicht positioniert **relative**: Positionierung relativ zur Normalposition **fixed**: Relativ zum Browser (Scroll hat keinen Effekt) **absolute**: Relativ zum nächsten positionierten Vorfahren **sticky**: basiert auf Scrollposition des Nutzers (wechselt zw. relative und fixed)	`div { position: static; } /* static (default), relative, absolute, fixed, sticky */`
z-index	Positionierte Elemente können überlappen → Stapelordnung	`div { position: absolute; z-index: 10; }`

Breite/ Hoehe

width	`div { width: auto; }`
min-width	`div { min-width: 20px; }`
max-width	`div { max-width: 100px; }`
height	`div { height: auto; }`
min-height	`div { min-height: 20px; }`
max-height	`div { max-height: 100px; }`

Anzeige

overflow	Anzeige von Elementen mit zu großem Inhalt **visible**: Inhalt ragt über Element hinaus **hidden**: Inhalt wird abgeschnitten **scroll**: Element bekommt Scrollleiste **auto**: Browser entscheidet automatisch	`div { overflow: hidden; }`
display	Definiert wie das Element angezeigt werden soll. Beispiele: **block**: Block Element **inline**: Inline Element **inline-block**: inline, aber Breite und Höhe definierbar **grid**: Grid Container **flex**: Flex Container	`div { display: block; }`

Übung 2

Erstelle ein HTML Dokument mit Grundgerüst. Der Body soll folgendes enthalten:

<div style=”width: 40px; border: 3px solid green;”>Dies ist ein Text, der viel zu lang für das div ist.

1. Schaue dir die Seite im Browser an. Was kann man beobachten?
2. Ändere den CSS Code für das Element so, dass der Text keine automatischen Zeilenumbrüche mehr enthält. Was kann man beobachten?
3. Positioniere das Element mittels CSS relativ zur Größe des Browserfensters 30% nach rechts und 30% nach unten.
4. Füge dem Element einen Innenabstand von 30px hinzu und beobachte was sich ändert.
5. Füge dem Element ein Außenabstand von 30px hinzu und beobachte was sich ändert.
6. Ändere das Element so, dass es eine Scrollleiste bekommt und man durch den Text scrollen kann.

Pseudo-Klassen + -Elemente

Pseudoklassen

Syntax: selector:pseudo-class { Eigenschaft: Wert; }

Beispiele für Pseudoklassen:

- link: Unbesuchter Link
- visited: Besuchter Link
- hover: Maus über den Link
- active: ausgewählter Link

```
a:hover { color: red; }
```

Pseudoelemente

```
p::first-line {
    color: #0000ff;
    font-variant: small-caps;
}
```

Syntax: selector::pseudo-element { Eigenschaft: Wert; }

Beispiele für Pseudoelemente:

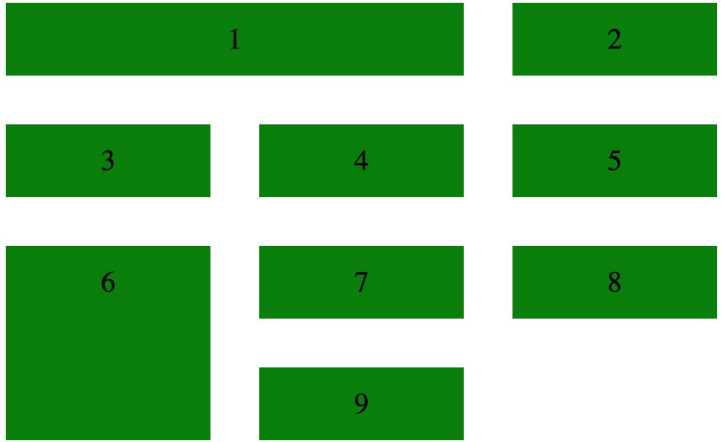
- First-line: Erste Zeile eines Texts
- First-letter: erster Buchstabe eines Texts
- Before: Einfügen von Inhalt vor selektierten Element(en)
- After: Einfügen von Inhalt nach selektierten Element(en)
- Marker: Formatierung von Markern vor List Item
- Selection: vom Benutzer ausgewählte Teile der selektierten Element(e)

Layout-Systeme

Grid

```html
<div class="container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
  <div class="item5">5</div>
  <div class="item6">6</div>
  <div class="item7">7</div>
  <div class="item8">8</div>
  <div class="item9">9</div>
</div>
```

```css
.container {
 display: grid;
 grid-template-columns: auto auto auto; /* three columns */
 column-gap: 50px;
 row-gap: 50px;
}
.container > div {
 background-color: green;
 text-align: center;
 padding: 20px 0;
 font-size: 30px;
}
.item1 {
 grid-column-start: 1; /* start at grid line 1 */
 grid-column-end: 3; /* end at grid line 3 */
}
.item6 {
 grid-row-start: 3; /* start at grid line 3 */
 grid-row-end: 5; /* end at grid line 5 */
}
```



Flexbox

```html
<div class="container">
    <div class="item item1">1</div>
    <div class="item item2">2</div>
    <div class="item item3">3</div>
    <div class="item item4">4</div>
    <div class="item item5">5</div>
    <div class="item item6">6</div>
    <div class="item item7">7</div>
    <div class="item item8">8</div>
    <div class="item item9">9</div>
</div>
```

```css
.container {
 display: flex;
 flex-direction: row; /* row (default) | row-reverse | column | column-reverse */
 flex-wrap: wrap; /* nowrap (default) | wrap | wrap-reverse */
 gap: 20px;
}
.container > div {
 background-color: red;
 width: 100px;
 text-align: center;
 line-height: 75px;
 font-size: 30px;
}
.item {
   flex-grow: 1; /* flex-grow will stretch the items to fill the container */
}
.item3 {
   flex-grow: 2; /* relative to flex-grow of rest of the items (default 0) */
}
```

Grid vs. Flexbox

- Flexbox needs less code
- Grid behaves more predictably
- Grid is more explicit
- → Grid for complex two dimensional layouts which are well defined
- → Flexbox for one dimensional layouts (only rows or columns, not mixed)

Übung 3

Baue folgendes Grid mit hilfe von CSS Grid nach:

<table>
<tr><td>1</td><td colspan="2">2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td rowspan="3">6</td><td>7</td></tr>
<tr><td rowspan="2">8</td><td>9</td><td>10</td></tr>
<tr><td>11</td><td>12</td></tr>
</table>

Assignment 1

https://classroom.github.com/a/vs0u1FVv

Erstellen Sie eine Website mit folgenden Pages:

- Welcome Page mit Bild und Beschreibung (z.B. Movie Blog)
  - Header Zeile mit Logo und Nav Bar (Klick auf Logo führt zurück zur Hauptseite)
  - Unterschiedliche HTML Sections (Link in Nav Bar)
  - Tabelle mit Content Auflistung und Link zu einer Detailseite (min. 3 Filme)
  - Kontaktformular (muss keine Funktion haben)
  - Footer mit Impressum
- Detail Page zu mindestens einem der Filme:
  - Überschrift, Story, Streamingdienst-Liste, Bild (mit Link zur Serie)
  - Fußzeile mit Link zurück die Hauptseite

Dabei sollen folgende Technologien eingesetzt werden:

- HTML nav und HTML section
- HTML table
- HTML input
- CSS Grid als Layout für die Pages
- CSS Definitionen in externes Style Sheet auslagern (.css Datei)