CSS -Cheatsheet

Inhalt

[Einbindung CSS in HTML 3](#_Toc785392)

[Externe CSS Datei im gleichen Verzeichnis 3](#_Toc785393)

[Externe CSS Datei als URL 3](#_Toc785394)

[Inline Styling mit CSS an einzelnen HTML Elementen 3](#_Toc785395)

[Inline Syling mit CSS mit eigener style-Sektion im Head 3](#_Toc785396)

[Auswählen einzelner HTML-Tags in externer CSS-Datei 3](#_Toc785397)

[Auswählen von Klassenattributen in CSS 3](#_Toc785398)

[Auswählen von mehreren Klassenattributen 4](#_Toc785399)

[Auswählen von HTML Elementen mit ID-Bezug 4](#_Toc785400)

[Überschreiben in CSS 4](#_Toc785401)

[Verknüpfen mehrerer CSS-Selektoren 4](#_Toc785402)

[Verschachtelte Elemente 5](#_Toc785403)

[Verkettung und Spezifität 5](#_Toc785404)

[!Important-Schlüsselwort 5](#_Toc785405)

[Mehrere Selektoren 6](#_Toc785406)

[Grundstruktur CSS 6](#_Toc785407)

[Verändern der Schriftart (default-Wert: Times New Roman) 6](#_Toc785408)

[Verändern der Schriftgröße 6](#_Toc785409)

[Verändern der Schriftstärke 6](#_Toc785410)

[Verändern der Textausrichtung (right, left, center) 6](#_Toc785411)

[Verändern der Farbe 6](#_Toc785412)

[Verändern der Sichtbarkeit (von 0-1; 1 = 100%) 7](#_Toc785413)

[Hintergrundbild als Hintergrund 7](#_Toc785414)

[Grid 7](#_Toc785415)

[Grid-template-columns 7](#_Toc785416)

[Grid-template-rows 7](#_Toc785417)

[Grid-template 7](#_Toc785418)

[Fraction 7](#_Toc785419)

[Repeat() 7](#_Toc785420)

[minmax() 7](#_Toc785421)

[grid-column-gap 8](#_Toc785422)

[grid-row-gap 8](#_Toc785423)

[grid-row-start, grid-row-end 8](#_Toc785424)

[grid-column-start, grid-column-end 8](#_Toc785425)

[grid-area 8](#_Toc785426)

[grid-teamplate-areas 8](#_Toc785427)

[justify-items 8](#_Toc785428)

[justify-content 9](#_Toc785429)

[align-items 9](#_Toc785430)

[align-content 9](#_Toc785431)

[justify-self 9](#_Toc785432)

[aling-self 9](#_Toc785433)

[grid-auto-row 9](#_Toc785434)

[grid-auto-columns 9](#_Toc785435)

[grid-auto-flow 9](#_Toc785436)

[Das Box Model 9](#_Toc785437)

[Höhe und Breite 9](#_Toc785438)

[Rahmen (=border) um Inhalte 10](#_Toc785439)

[Rahmenform 10](#_Toc785440)

[Abstand zwischen Inhalt und Rahmen einer Box 10](#_Toc785441)

[Abstände außerhalb der Box 10](#_Toc785442)

[Zentralisierung von Inhalten 11](#_Toc785443)

[Vertikale Margins brechen zusammen 11](#_Toc785444)

[Begrenzung der Breite und Höhe von Elementen beim Zoomen 11](#_Toc785445)

[Überlappung von Inhalten 11](#_Toc785446)

[Zurücksetzen auf Default Werte 12](#_Toc785447)

[Sichtbarkeit einzelner Elemente (visible and hidden) 12](#_Toc785448)

[Fixe Höhe und Breite des Boxmodels 12](#_Toc785449)

[Problem: Elemente nehmen gesamte horizontale Breite der Webseite ein 12](#_Toc785450)

[z-index-property 13](#_Toc785451)

[display-property 13](#_Toc785452)

[Float-Property 13](#_Toc785453)

[Clear-Property 14](#_Toc785454)

[Farben in CSS 14](#_Toc785455)

[Vordergrund vs. Hintegrund 14](#_Toc785456)

[Hexadezimalfarben 14](#_Toc785457)

[RGB-Farben 15](#_Toc785458)

[HSL-Farben/ Hue-Saturation-Lightness-Colors 15](#_Toc785459)

[Undurchlässigkeit 15](#_Toc785460)

[Typographie 15](#_Toc785461)

# Einbindung CSS in HTML

## Externe CSS Datei im gleichen Verzeichnis

* <head>
  + <link href="./style.css" type="text/css" rel="stylesheet">
* </head>

## Externe CSS Datei als URL

* <head>
  + <link href=<https://www.codecademy.com/stylesheets/style.css> type=”text.css” rel=”stylesheet”>
* </head>

## Inline Styling mit CSS an einzelnen HTML Elementen

* <p style="font-family:arial;">Hallo</p>

## Inline Syling mit CSS mit eigener style-Sektion im Head

* <head>
  + <style>
  + p{

font-family:arial;

}

* + </style>
* </head>

## Auswählen einzelner HTML-Tags in externer CSS-Datei

* h1{

color: maroon;

}

## Auswählen von Klassenattributen in CSS

* HTML Datei:
  + <h1 class="title">Top Vacation Spots</h1>
* CSS Datei:
  + .title{

color:teal;

}

## Auswählen von mehreren Klassenattributen

* HTML Datei:
  + <h1 class="title uppercase">Top Vacation Spots</h1>
* CSS Datei:
  + .title{

color:teal;

}

* + .uppercase{

text-transform: uppercase;

}

## Auswählen von HTML Elementen mit ID-Bezug

* HTML Datei
  + <h1 class="title uppercase" id="article-title">Top Vacation Spots</h1>
* CSS Datei
* #article-title{

font-family: cursive;

text-transform: capitalize;

}

## Überschreiben in CSS

* Spezifitätsrangfolge (von hoch nach niedrig)

1. ID
2. Klassen
3. Tags

* Je höher die Spezifität, desto leichter ist es zu überschreiben.
* HTML:
  + <h1 class=“headline“>Breakline</h1>
* CSS:
  + h1{

color:red;

}

* + .headline{

color:firebrick;

}

* Die Überschrift Breakline würde in der Farbe „firebrick“ erscheinen, da Klassen spezifischer als Tags sind.

## Verknüpfen mehrerer CSS-Selektoren

* CSS:
  + h2.destination{

font-family:cursive;

}

* Es werden nur h2-Elemente mit der Klasse .destination gestylt, jedoch z.B. keine p-Elemente mit der Klasse .destination.

## Verschachtelte Elemente

* HTML:
* <h5>Top Attractions</h5>

<ul>

<li>Museums</li>

<li>Bike Tours</li>

<li>Historical Monuments</li>

</ul>

<div class="description">

* CSS:
  + .description h5 {

color: teal;

}

## Verkettung und Spezifität

* HTML:
  + <h5>By: Stacy Gray</h5>
  + <div class="description">
  + <h5>Top Attractions</h5>
  + </div>
* CSS:
  + .description h5 {

color: teal;

}

* + h5{

color: rebeccapurple;

}

* Die Überschrift Stacy Gray erscheint in der Farbe „rebeccapurple“, die h5 Überschriften der Klasse description erscheinen in „teal“.

## !Important-Schlüsselwort

* CSS:
  + .description h5 {

color: teal;

}

* + h5{

color: rebeccapurple !important;

}

* Durch das Schlüsselwort “!important” erscheinen alle h5-Überschriften in der Farbe rebeccapurple. Spezifischere Selektoren werden hierdruch nicht beachtet.

## Mehrere Selektoren

* CSS:
  + anstatt:
    - h5{

font-family: Georgia;

}

* + - p{

font-family: Georgia;

}

* + besser: (redundanten Code vermeiden: h5 und p erhalten Schriftart Georgia)
    - h5, p{

font-family: Georgia;

}

## Grundstruktur CSS

* selected element{

property: value;

}

* h1{

color: blue;

}

## Verändern der Schriftart (default-Wert: Times New Roman)

* p{

font-family: Helvetica;

}

## Verändern der Schriftgröße

* p{

font-size: 18px;

}

## Verändern der Schriftstärke

* p{

font-weight: bold;

}

## Verändern der Textausrichtung (right, left, center)

* h1{

text-align: center;

}

## Verändern der Farbe

* Hintegrrundfarbe = background-color
* Schriftfarbe = color
* .caption{

background-color: white;

color: white;

}

## Verändern der Sichtbarkeit (von 0-1; 1 = 100%)

* .caption{

opacity: 0.75;

}

## Hintergrundbild als Hintergrund

* .image {

background-image: url("https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-2/soccer.jpeg");

}

# Grid

* Grid container <div class=”grid”>
* Benutzen von display:grid in css zum veranschaulichen

# Grid-template-columns

* grid-template-columns: 100px 200px;
* erstellt in dem grid 2 Spalten, eine mit 100px breite und eine mit 200px
* kann auch mit % werten arbeiten

# Grid-template-rows

* grid-template-rows: 100px 20% 600px;
* setzt Höhe von den jeweiligen Zeilen

# Grid-template

* vereint grid-template-rows und -columns
* grid-template: 200px 300px / 20% 10% 70%; (zuerst Zeilen dann Spalten)

# Fraction

* neue Einheit fr
* teilt breite und höhe von Grid in Teile

# Repeat()

* repeat(2,100px) => macht 2 mal 100px column oder row

# minmax()

* minmax(100px,500px)
* column oder row wird nie unter 100px oder über 500px gehen

# grid-column-gap

# grid-row-gap

* grid-column-gap: 10px;
* zwischen den grids wird platz von 10px gelassen
* grid-gap: 20px 10px;
* setzt Zeilen auf 20px Abstand und Spalten auf 10px

# grid-row-start, grid-row-end

* man kann auswählen wie viele rows und ein element besetzten soll
* grid-row-start:1;
* grid-row-end:3;
* besetzt row 1 und 2
* grid-row: 4/6; -> abgekürzt, Start bei 4 und Ende bei 5

# grid-column-start, grid-column-end

* man kann auswählen wie viele columns und ein element besetzten soll
* grid-column: 4 / span 2;
* grid-column: 4/ 6; -> das gleiche, nur dass span 2 bedeutet, es soll 2 Zeilen besetzten

# grid-area

* setzt Start und Ende von column und row
* grid-area: 2 / 3 / 4 /span;
* zuerst row-start,column start,row-end, column-end

# grid-teamplate-areas

* kann man grid system einrichten mit den Namen, die angesprochen werden sollen
* grid-template-areas: “head head “

“nav nav “

* 2 rows 2 columns mit den Namen
* Die kann man dnan elementen übergeben in css
* header{

grid-area:head;

}

* mithilfe von z-index können Elemente overlapped werden

# justify-items

* positioniert grid elemente entlang der row-Achse (x-Achse)
* Werte sind start,end,center,stretch

# justify-content

* Positionieren eines kompletten grids entlang der Zeilen-Achse
* Justify-content: start | end | center | stretch | space-around | space-between | space-evenly;

# align-items

* Positioniert grid items entlang der Spalten-Achse
* align-items: start | end | center | stretch

# align-content

* positioniert grid entlang der Spalten-Achse
* align-content: start | end| center| stretch| space-around| space-between| space-evenly;

# justify-self

# aling-self

* spezifiziert wie sich ein Objekt entlang der x bzw. y-Achse Objekte positionieren
* justify-self: start | end| center| stretch
* überschreibt -items properties

# grid-auto-row

# grid-auto-columns

* bei zu wenig grids, bei dem eigentlich mehr grid plätze benötigt werden um alles anzuzeigen, wird ein Algorithmus angewandt, der mehr grid plätze hinzufügt
* mit grid-auto-row kann man die Höhe der hinzugefügten grids angeben
* mit grid-auto-columns kann man nur die Breite angeben
* mögliche Werte: px, %, fr , repeat()

# grid-auto-flow

* gibt an, ob in der Zeile oder Spalte neue Elemente hinzugefügt werden sollen
* grid-auto-flow: row | column | dense

# Das Box Model

## Höhe und Breite

* Inhalte von Elementen haben 2 Dimesnionen: Höhe (height) und Breite (width)
* Durch Verändern der default Werte in Pixeln kann die exakte Größe der Elemente gesetzt werden
* p{

width:240px;

height: 80px;

}

## Rahmen (=border) um Inhalte

* border: width style color; Rahmen: Dicke Aussehen Farbe;
* p{

border: 3px solid coral;

}

* style: none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset

## Rahmenform

* h2{

border: dotted red;

border-radius: 15px;

}

* durch border-radius abgerundeter Rahmen, nicht mehr quadratisch
* durch 100% als Wert von border-radius -> kreisförmiger Rahmen

## Abstand zwischen Inhalt und Rahmen einer Box

* alle Abstände gleich
* p{

border: 3px solid coral;

padding: 10px;

}

* Abstände einzeln festlegen
* p{

border: 3px solid coral;

padding-bottom: 20px;

padding-top: 20px;

padding-left: 30px;

padding-right: 30px;

}

* Abstände einzeln festlegen (Kurzschreibweise)
* p{

padding: top right bottom left;

}

* Bottom/Top-Abstände gleich und Left/Right Abstände gleich
* p{

padding: top/bottom left/right;

}

## Abstände außerhalb der Box

* alle Abstände gleich festlegen
  + p{

border: 1px solid aquamarine

margin: 20px;

}

* alle Abstände gleich festlegen
  + p{

margin-right: 20px;

margin-left: 20px;

margin-top: 30px;

margin-bottom: 30px;

}

* + p{

margin: top right bottom left

}

* rechts/ links und oben/unten gleiche Abstände
  + p{

margin: top/bottom left/right;

}

## Zentralisierung von Inhalten

* div.headline{

width: 400px;

margin: 0 auto;

}

* Ohne Setzen der width-Property Inanspruchnahme des Full-Screens und daher keine Zentralisierung möglich

## Vertikale Margins brechen zusammen

* #img-one{

margin-bottom: 30px;

}

* #img-two{

margin-top: 20px;

}

* margin vertikal = 30px => keine Addition wie bei horizuontalen Margins

## Begrenzung der Breite und Höhe von Elementen beim Zoomen

* p{

min-width: 300px;

max-width: 600px;

min-height: 150px;

max-height: 300px;

}

## Überlappung von Inhalten

* p{

overflow: scroll;

overflow: hidden;

overflow: visible;

}

* scroll -> Scrollbalken für überlappenden Inhalt
* hidden -> überlappender Inhalt unsichtbar
* visible -> überlappender Inhalt außerhalb des anderen Inhalts sichtbar

## Zurücksetzen auf Default Werte

* \* {

margin: 0;

padding: 0;

}

## Sichtbarkeit einzelner Elemente (visible and hidden)

* .navigation{

visibility: hidden;

}

## Fixe Höhe und Breite des Boxmodels

* durch Padding und Border verändern sich je nachdem festgelegte Höhe und Breite des Boxmodels
* Reset des ganzen Boxmodels:
  + \* {

box-sizing: border-box;

}

* + h1{

border: 1px solid black;

height: 200px;

width: 300px;

padding: 10px;

}

* Breite und Höhe nun durch Border thickness und padding nicht veränderbar

## Problem: Elemente nehmen gesamte horizontale Breite der Webseite ein

* Lösung: position-property mit den Werten
  + static
  + relative
  + absolute
* static
  + default Wert: links oben Position
* relative
  + top: bewegt Element nach unten
  + bottom: bewegt Element nach oben
  + right: bewegt Element nach links
  + left: bewegt Element nach rechts
  + .box-bottom{

position: relative;

top: 20px;

left: 50px;

}

* absolute
  + alle anderen Elemente werden absolute-Element ignorieren
  + Element wird relativ zum nächst gelegenen Elternelement platziert
  + header{

position: absolute;

width: 100%;

}

* fixed
  + Element auch beim Scrollen sichtbar
  + nur wenn anderes Element relative ist, ist fixed-Element ausgeblendet
  + h1{

position: fixed;

}

## z-index-property

* Rangfolge, inwiefern Elemente andere Elemente überlappen sollen
* Element mit hohem Index überlappt Element mit niedrigem Index
* header{

position: fixed;

width: 100%;

z-index: 10;

}

## display-property

* 3 mögliche Werte:
  + inline
    - Inhalte teilen sich horizontalen Platz mit anderen Elementen
    - default: <a>, <strong>, <em>
    - Inhalte in einer Zeile
    - h1{

display: inline;

}

* + block
    - Elemente sind nicht in der gleichen Zeile, wie der Inhalt drum herum
    - Elemente als default füllen ganze Breite der Seite aus
    - Block-Elemente durch default: h1-h1, p, div, footer
    - strong{

display: block;

}

* + inline-block
    - Kombination von inline und block-Werten
    - default: Bilder
    - Höhe und Breite können gesetzt werden
    - solange Breite es erlaubt, erscheinen Elemente nebeneinander
    - .answer{

display: inline-block;

width: 80px;

}

## Float-Property

* Elemente, soweit wie möglich rechts oder links positionieren mit float-property
* nur bei Elementen mit Position: relative oder static möglich
* .answer{

float: right;

}

## Clear-Property

* float-Property kann für mehrere Elemente verwendet werden
* Elemente können zusammenstoßen
  + Regelung des Verhaltens der Elemente bei Zusammenstoß durch clear-Property
* clear-Werte
  + left: auf der linken Seite keine Berührung möglich
  + right: auf der rechten Seite keine Berührung möglich
  + both: Berührung nicht möglich
  + none: Berührung auf beiden Seiten möglich
* div{

width: 200px;

float: left;

}

div.special{

clear: left;

}

## Farben in CSS

* 3 Möglichkeiten:
  + beschreibende Farben im Englischen
  + RGB-Farben: Zahlenwerte beschreiben Mix aus rot, grün und blau
  + HSL: Zahlenwerte beschreiben Mix aus Farbton, Sättigung und Helligkeit

### Vordergrund vs. Hintegrund

* color-property: stylt Element selbst
* background-color-property: stylt Hintegrund des Elements
* h1{

color: MidnightBlue;

background-color: Aqua;

}

### Hexadezimalfarben

* äquivalent zu beschriebenen Farben
* Tabelle:
  + DarkseaGreen #8FBC8F
  + Sienna #A0522D
  + SaddleBrown #8B4513
  + Brown #A52A2A
  + Black #000000 oder #000
  + White #FFFFFF oder #FFF
  + Aqua0FF #00FFFF oder #0FF

### RGB-Farben

* h1{

color: **rgb**(23, 45, 23);

}

* rgb(rot, grün, blau)
* Zahlenwerte von 0-255

### HSL-Farben/ Hue-Saturation-Lightness-Colors

* color: **hsl**(hue, saturation, lightness);
* color: **hsl**(120, 60%, 70%);
* hue = Farbton von 0-360:
  + 0 = rot
  + 120 = grün
  + 240 = blau

### Undurchlässigkeit

* alpha/ opacity als vierter Wert bei rgb und hsl
* rgb -> rgba; hsl -> hsla
* .midground {

background-color: **hsla**(225, 100%, 25%, 0.4);

}

* body {

background-color: **rgba**(0, 255, 0, 0.1);

}

* zero-transparency durch:
  + color: tranparent;

### Typographie

#### Schriftart:

* + p{

font-family: Helvetica;

}

#### Schriftstärke I

* + p{

font-weight: bold;

}

#### Schriftstärke II

* Zahlenwerte für font-weight erlaubt
  + valide Werte: 100-900 in 100er-Schritten
  + 400: default
  + 700: bold
  + 300: light

#### Schriftaussehen

* kursive Schrift
* .font-card .creator {

font-style: **italic**;

}

#### Abstand zwischen Wörtern

* h1{

word-spacing: 0.3em;

}

#### Abstand zwischen Buchstaben

* h1{

letter-spacing: 0.02em;

}

#### Texttransformierung

* ständige Großschreibung
  + h1{

text-transform: uppercase;

}

* ständige Kleinschreibung
  + h1{

text-transform: lowercase;

}

#### Textausrichtung

* Werte für text-align:
  + right, center, left
* h1{

text-align: right;

}

#### Zeilenabstand

* p{

line-height: 1.4;

}

#### Schriftart, wenn andere Schriftart nicht verfügbar

* .garamond .sample .text{

font-family: "Garamond", serif;

}

#### Schriftarten einfügen bzw auf Schriftarten verlinken I

* im Head der HTML-Datei:
  + <link href="https://fonts.googleapis.com/css?family=Space+Mono:400,700" rel="stylesheet">
* in CSS:
  + .space .sample .text {

font-family: "Space Mono", monospace;

}

#### Schriftarten einfügen bzw auf Schriftarten verlinken II

* @font-face {

font-family: 'Space Mono';

src: local('Space Mono'), local('SpaceMono-Regular'), url(https://fonts.gstatic.com/s/spacemono/v1/adVweg3BJhE6r8jYmXseHQzyDMXhdD8sAj6OAJTFsBI.woff2) format('woff2');

unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02C6, U+02DA, U+02DC, U+2000-206F, U+2074, U+20AC, U+2212, U+2215;

}

#### Lokale Schriftarten hinzufügen

* @font-face {

font-family: "Glegoo";

src: url(../fonts/Glegoo-Regular.ttf) format('truetype');

}