CSS -Cheatsheet

Inhalt

[Einbindung CSS in HTML 2](#_Toc507194)

[Externe CSS Datei im gleichen Verzeichnis 2](#_Toc507195)

[Externe CSS Datei als URL 2](#_Toc507196)

[Inline Styling mit CSS an einzelnen HTML Elementen 2](#_Toc507197)

[Inline Syling mit CSS mit eigener style-Sektion im Head 2](#_Toc507198)

[Auswählen einzelner HTML-Tags in externer CSS-Datei 2](#_Toc507199)

[Auswählen von Klassenattributen in CSS 2](#_Toc507200)

[Auswählen von mehreren Klassenattributen 2](#_Toc507201)

[Auswählen von HTML Elementen mit ID-Bezug 3](#_Toc507202)

[Überschreiben in CSS 3](#_Toc507203)

[Verknüpfen mehrerer CSS-Selektoren 3](#_Toc507204)

[Verschachtelte Elemente 4](#_Toc507205)

[Verkettung und Sepzifität 4](#_Toc507206)

[!Important-Schlüsselwort 4](#_Toc507207)

[Mehrere Selektoren 5](#_Toc507208)

[Grundstruktur CSS 5](#_Toc507209)

[Verändern der Schriftart (default-Wert: Times New Roman) 5](#_Toc507210)

[Verändern der Schriftgröße 5](#_Toc507211)

[Verändern der Schriftstärke 5](#_Toc507212)

[Verändern der Textausrichtung (right, left, center) 5](#_Toc507213)

[Verändern der Farbe 5](#_Toc507214)

[Verändern der Sichtbarkeit (von 0-1; 1 = 100%) 6](#_Toc507215)

[Hintergrundbild als Hintergrund 6](#_Toc507216)

# 

# 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