CSS – Cascading Style Sheets

CSS ist wie HTML keine Programmiersprache. Es ist auch keine Markup-Sprache wie HTML, sondern eben eine Stylesheet-Sprache, die es erlaubt für Elemente auf der Seite das Aussehen festzulegen. Man benötigt zum Beispiel folgenden Code um alle Absätze <p> einer Seit rot darzustellen.

```
p {
  color: red;
}
```

Wie bindet man CSS in das HTML-Dokument ein?

Hierfür gibt es mehrere Möglichkeiten. Je nach Anwendungsfall kann die Einbindung unterschiedlich erfolgen.

1. Mittels externer CSS-Datei
   Diese Methode ist die erste Wahl, wenn die enthaltenen Styles für mehrere HTML-Seiten zutreffend sind.
   Direkt im <head> der HTML-Datei wird auf das Stylesheet-File referenziert. Bei jedem Seitenaufruf wir das File abgerufen und eingebunden.

   Achtung: Der Browser-Cache kann eine Aktualisierung verhindern, da bereits geladene CSS-Dateien direkt aus dem Browser-Cache geladen werden. Vorteil ist, dass die Seite schneller geladen wird. Nachteil ist, dass die Formatierung nicht gleich übernommen wird.
   Praxistipp: Mit F5 kann man die gesamte Seite neu laden. Mit Strg+F5 hingegen wird vor dem Laden der Browser Cache geleert.

   Beispiel: (für eine externe CSS-Datei)

```
2   <html lang="de">
3   <head>    <meta charset="utf-8">
4       <meta http-equiv="X-UA-Compatible" content="IE=edge">
5       <meta name="viewport" content="width=device-width, initia
6       <!-- Die 3 Meta-Tags oben *müssen* zuerst im head stehen;
7       <title>Bootstrap-Basis-Vorlage</title>
8       <!-- Bootstrap -->
9       <link href="css/bootstrap.min.css" rel="stylesheet">
10      <link href="css/myStyles.css" rel="stylesheet">
```

   rel definiert den logischen Beziehungstyp des Elements. In diesem Beispiel wird ein Stylesheet eingebunden.
   href referenziert die einzubindende Stylesheet-Datei.

2. Style-Tag
   Wenn im Vorhinein klar ist, dass die Stylesheet-Angabe nur auf eine Seite angewandt werden, kann man die Stylesheet-Angaben direkt im HTML-Dokument ablegen. Die Style-Angaben sind hier ein Kind-Element des Head-Elements.

Beispiel: (Style-Tag im HTML-Dokument Eigenschaften überschreiben)

```
<head>    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Die 3 Meta-Tags oben *müssen* zuerst im head stehen; jeglicher sonstiger
    <title>Bootstrap-Basis-Vorlage</title>
    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <link href="css/myStyles.css" rel="stylesheet">
    <style>
        li {
            color: #86cfda;
        }
    </style>
</head>
</head>
```

3. Inline im HTML-Tag
   Das HTML-Tag stellt die kleinste Einheit für die Angabe von CSS-Formatierungen dar.
   *<**p** style="text-align: center; color: green;">*
   Inline-Styles haben immer die höchste Wertigkeit und überschreiben Styles aus anderen Quellen.

```
<h1 style="text-align: center; color: green;">Hello, world!</h1>
```

4. @Import
   <style> @import url("style.css") screen; </style>

   @import ist keine empfehlenswerte Methode zum Einbinden einer CSS-Datei, denn importierte CSS-Dateien blockieren parallele Downloads. Der Web-Browser muss warten, bis die importierte Datei geladen ist, bevor er weitere Dateien herunterladen kann. Sinnvoll ist @import allenfalls, wenn sichergestellt werden soll, dass bestimmte CSS-Eigenschaften vor anderen Eigenschaften geladen werden.

Das Style-Tag und externe CSS-Dateien sollten am Beginn des <head> eingebunden werden, da sie der Darstellung der Seite dienen. Andernfalls würden sich die Style-Angaben erst nach dem vollständigen Laden der Seite auswirken.

Kaskadierung

Da sich CSS-Eigenschaften gegenseitig überschreiben können, entsteht die CSS-Kaskade – daher der Begriff *Cascading Stylesheets*. Je näher die Regel am HTML-Element liegt desto höher wird Sie gewichtet. Die Überschreibung von CSS-Regeln wird oftmals bei der Verwendung von Website-Frameworks wie Bootstrap eingesetzt, um Standards wie z. B. die Hauptfarbe auf die eigenen Bedürfnisse anzupassen.

Folgende Reihenfolge wird für die Kaskadierung herangezogen:

1. @import
2. Externe CSS-Files
3. Style-Tag im Html Dokument
4. Inline-Styles (HTML-Tag)

Z. B. würde die Eigenschaft „color“ für den <h1>-Tag in einer externen Datei und inline definiert worden sein, gewinnt die Inline-Regel. Außer man würde in der externen Datei die Eigenschaft mit dem Attribute „!important“ kennzeichnen.

Aufbau von CSS-Regeln

Beispiel:

p=Selektor auf HTML-Tag-Ebene
color=Eigenschaft/property
red=Eigenschaftswert/value

Die ganze Struktur wird Regelsatz (oder oft nur Regel) genannt.

Um mehrere Eigenschaften eines HTML-Elements auf einmal zu verändern, trennt man die Deklarationen innerhalb eines Regelsatzes mit Semikolons, wie folgt:

```
1   p {
2       color: red;
3       font-size: 16px;
4   }
```

Man kann auch mehrere Elemente gleichzeitig auswählen:

```
1   p, li, h1 {
2       color: red;
3       font-size: 16px;
4   }
```

Es gibt mehrere verschiedene Arten von Selektoren. In der folgenden Tabelle sind einige weitere Arten dargestellt, um spezifischere Selektoren zu nutzen.

Selector name	Was wird ausgewählt?	Beispiel
Element- Selektor (auch Tag- oder Typ-Selektor genannt)	Alle HTML-Elemente eines bestimmten Typs.	p Wählt alle <p>-Elemente aus.

ID-Selektor	Element mit der entsprechenden ID wird ausgewählt. (Eine ID kann immer nur einem einzigen Element innerhalb eines Dokuments zugeordnet werden)	`#my-id` `Wählt <p id="my-id">` oder `<a id="my-id">` `aus.`
Klassen-Selektor	Element(e) mit der entsprechenden Klasse werden ausgewählt. (Klassen können mehreren Elementen innerhalb eines Dokuments zugeordnet werden)	`.my-class` `Wählt <p class="my-class">` und `<a class="my-class">` `aus.`
Attribut-Selektor	Element(e) mit entsprechendem Attribut werden ausgewählt.	`img[src]` `Wählt <img src="myimage.png">` , aber nicht `<img> aus.`
Pseudoklassen-Selektoren	Element(e) eines bestimmten Typs, welche sich in einem bestimmten Zustand befinden (z.B.: Mauszeiger ist über dem Element)	`a:hover` `Wählt <a>` nur dann aus, wenn der Mauszeiger darüber bewegt wird.

Es gibt viele weitere Selektoren. Siehe Quelle:(https://developer.mozilla.org/de/docs/Web/Guide/CSS/Getting_started/Selectors)

Ein weiteres Beispiel: Wir setzen für <h1> <p> Schriftgrößen, Linienhöhen und einen Buchstabenabstand.

```css
h1 {
  font-size: 60px;
  text-align: center;
}

p, li {
  font-size: 16px;
  line-height: 2;
  letter-spacing: 1px;
}
```

Die bisherigen Beispiele liefern nur einen kleinen Auszug der CSS-Properties. Weiterführende Properties kann man z. B. unter https://www.w3schools.com/css/default.asp nachlesen. Auch bei der Verwendung von Bootstrap-Komponenten hat man immer wieder Berührungspunkte mit CSS-Properties.

CSS wird nicht nur für das Aussehen von HTML-Seiten verwendet, sondern auch für das Aussehen des Druckbildes. Dies kann mit sogenannten Media-Queries umgesetzt werden, indem man durch die zusätzliche Angabe des ”Mediums” mittels @media print ein gesonderten Style für das jeweilige Medium – hier für den Drucker – angibt. Darüber hinaus werden Media-Queries auch für die responsive Darstellung von Webseiten verwendet (= unterschiedliche Darstellung auf den einzelnen Gerätetypen oder Bildschirmgrößen).

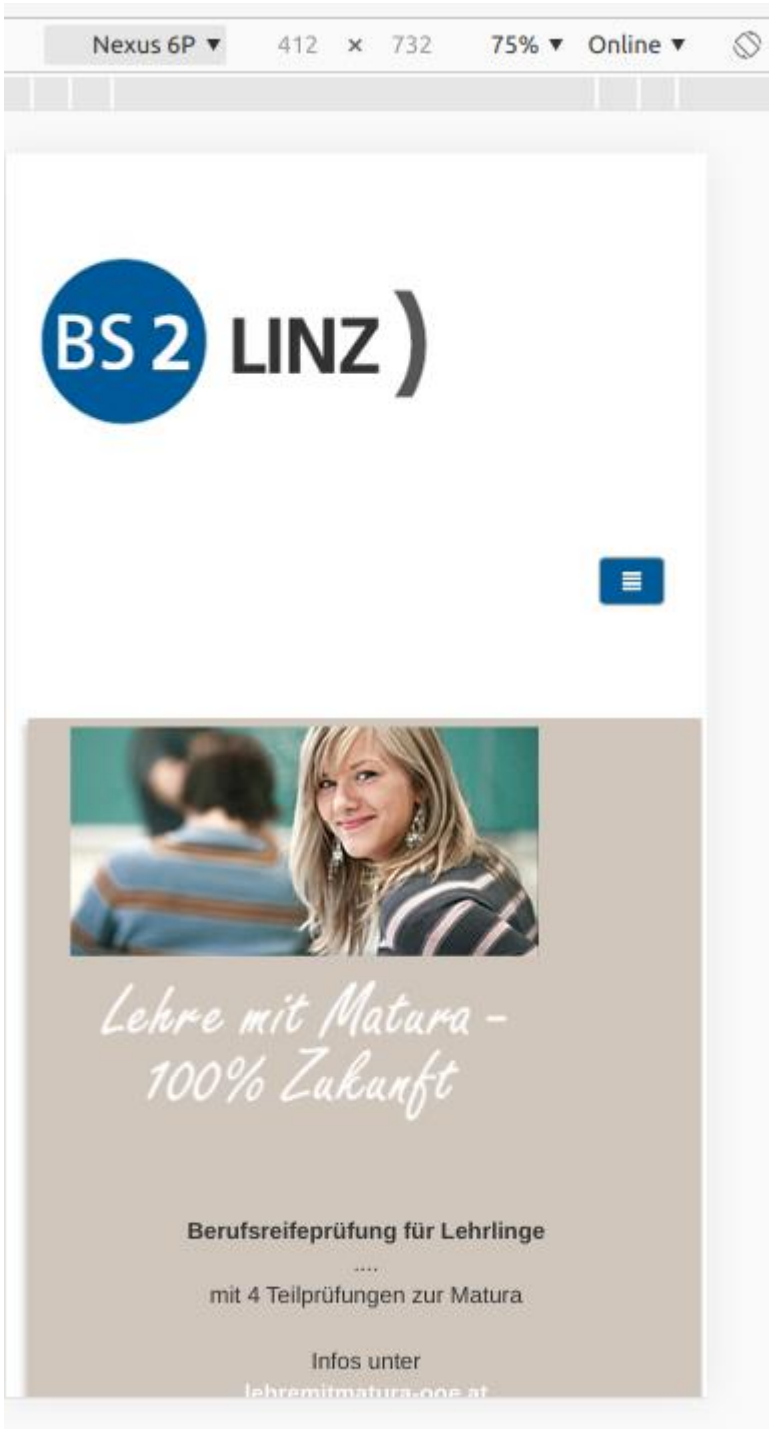
```html
<style>
    li {
        color: #86cfda;
    }
    /* Set the background color of body to tan */
    body {
        background-color: tan;
    }
    /* On screens that are 992px or less, set the background color to blue */
    @media screen and (max-width: 992px) {
        body {
            background-color: blue;
        }
    }
    /* On screens that are 600px or less, set the background color to olive */
    @media screen and (max-width: 600px) {
        body {
            background-color: olive;
        }
    }
</style>
```

TIPP: Die Bildschirmgröße oder das Device lässt sich im Webbrowser simulieren. Diese Möglichkeit

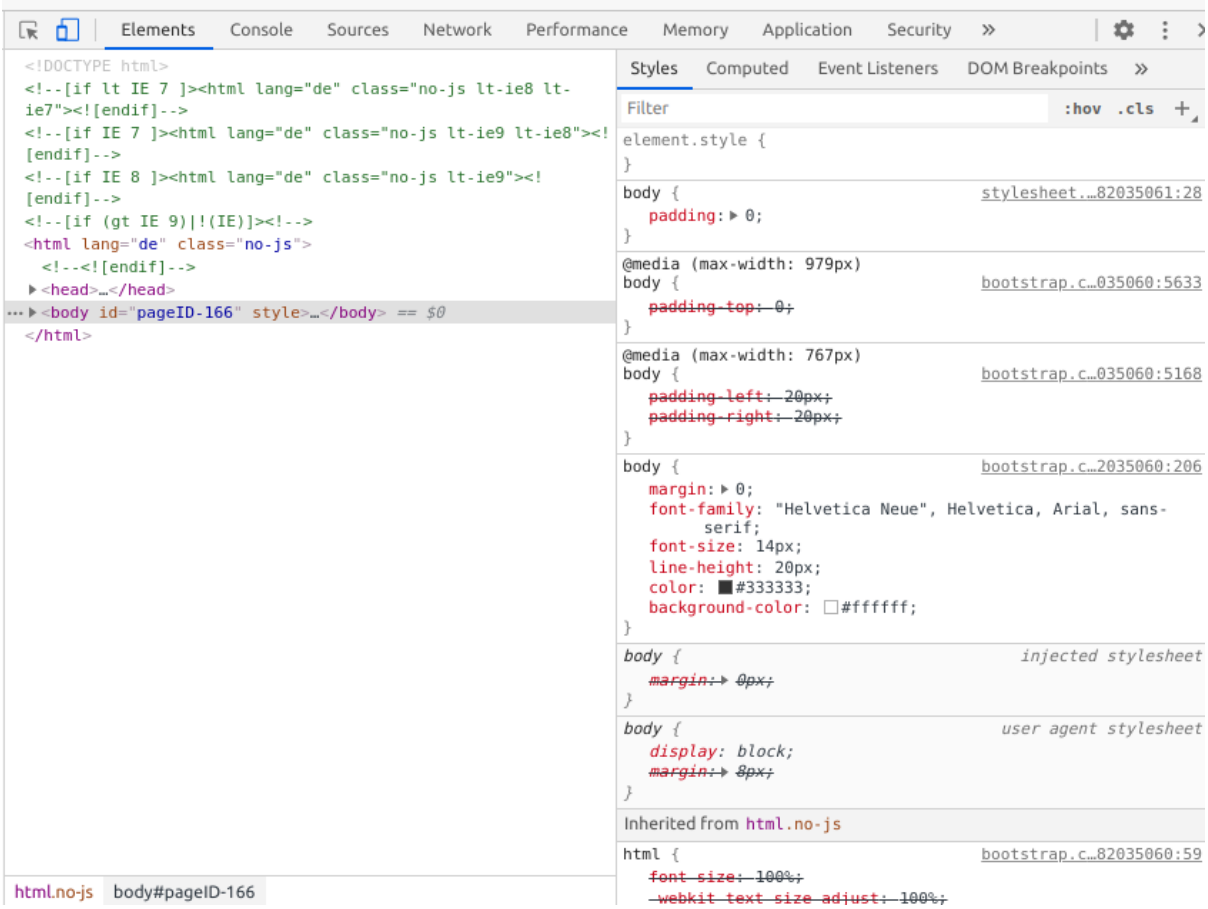
kann den Testaufwand von Webseiten erheblich verringern.

Chrome Dev-Tools:

Webseite im Browser öffnen -> F12 Dev-Tools aufrufen --> Toggle Device Mode einschalten -> Anzeigegerät auswählen.



Die Webseite wurde mit einem veralteten Smartphone (Nexus 6P) simuliert. In den Dev-Tools sieht man zusätzlich welche CSS-Regeln zu Anwendung kommen.



Hier kommen Media-Queries in Bezug auf die Pixelweite des Device (Nexus 6p) zur Anwendung.

Grundsätzlich sind die Dev-Tools (F12 in Chrome- oder Firefox-Browser) ein guter Weg, um sich mit den CSS-Properties vertraut zu machen Die einzelnen Properties lassen sich auch verändern, die Auswirkungen sind on-the-fly ersichtlich. In den Dev-Tools kann sowohl der HTML-Code (Elements) als auch die dahinterliegenden CSS-Regeln (Styles) einer Seite eingesehen und direkt abgeändert werden.

Anmerkung zu CSS: Die Einbindung von großen externen CSS-Dateien kann die Seitenladezeit negativ beeinträchtigen. Mit CDNs – Content-Delivery-Networks kann man die Lastverteilung verbessern und somit die Performance steigern. CSS-Files für Bootstrap können über eine externe Url eingebunden werden.
https://cdnjs.cloudflare.com/ajax/libs/bootstrap-daterangepicker/3.0.5/daterangepicker.css

Responsive Web-Design

Mit responsive Web-Design sorgt man dafür, dass Webseiten auf unterschiedlichsten Geräten wie Desktop-Computer, Laptop, Table, Smartphone, Smart-Tv, … in einem ansprechenden Design dargestellt werden. Responsive Web-Design wird durch die Kombination von CSS und HTML ermöglicht.
Eine gute Quelle für eine ausführliche Beschreibung:
https://www.w3schools.com/css/css_rwd_intro.asp

Bevor es Smartphones und Tablets gab, wurden Webseiten mit einem statischen Design und mit fixen Größen programmiert. Mit dem Aufkommen von Tablets und Smartphones, welche eine niedrigere Auflösung als Monitore hatten, kam das Problem auf, dass Webseiten auf Smartphones schlecht zu bedienen waren. In einer Übergangsphase begann man zusätzlich zur “normalen” Webseite “mobile” Webseiten zu entwickeln, diese waren üblicherweise über eine Subdomain m.domain.at erreichbar. Ein wesentlicher Nachteil waren die hohen Entwicklungs- und Wartungskosten. Änderungen mussten immer mindestens an zwei “Webseiten” vorgenommen werden.

Viewport

Durch die Einführung des Viewports mit HTML5 wurde responsive Webdesign möglich. Der Viewport ist der für den User sichtbare Bereich der Webseite. Dieser Bereich kann abhängig vom Gerät, auf dem die Webseite angezeigt wird, unterschiedlich groß sein.

Durch die Angabe des HTML meta-tags “viewport” content=”width=device-width, initial-scale=1.0” wird dem Browser des aufrufenden Gerätes mitgeteilt, dass er zur Darstellung die maximale Größe des “ausführenden” Gerätes verwenden soll. Sämtliche Inhalte werden dadurch skaliert. Die Angabe “width=device-width” steuert, dass die maximale Breite des Gerätes genutzt werden soll. Die Angabe “initial-scale=1” steuert den Zoom-Level für die erste Anzeige (in diesem Fall 100%).

```
3   <head>
4       <meta charset="utf-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1">
```