# Web Application Development

Cascading Style Sheets

Martina Freundorfer

## Letztes Mal

- **Hyper Text Markup Language**
- Geschichtliches: HTML x (  … 2, 3, 4)
- XHTML – HTML basierend auf XML
- HTML 5 - W3C Recommendation seit Oktober 2014, aktuell bei 5.2
- WHATWG – Living Standard for HTML (5 wird weggelassen)

  Nachtrag: WHATWG vs. W3C - Die WHATWG setzt das W3C nicht außer Kraft, da die vorgeschlagenen Standards, die von der WHATWG geschaffen werden, beim W3C zur Zustimmung oder Nachbesserung eingereicht werden. Zukünftig plant die WHATWG, noch enger mit dem W3C zusammenzuarbeiten.
- HTML-Dokument als Baum mit “Wurzel und Blättern” (root and elements)
- **Mit HTML werden Inhalte strukturiert.**
- https://www.w3schools.com/quiztest/quiztest.asp?qtest=HTML

# Semesterplan

| # | KW | Datum | Vorlesung | **Das Labor nach der Vorlesung findet für beide Gruppen wöchentlich statt.** |
|---|---|---|---|---|
| 1 | 14 | 3.4. | Einführung, Scheinkriterien | npm im Selbststudium, Anlegen eines Webprojektes, Installation von Live Server |
| 2 | 16 | 17.4. | Client-Server, Web Apps, URI, HTTP | Aufgabe von Beleg 1: AdViz (nur HTML und CSS) |
| 3 | 17 | 24.4. | HTML, CSS | Prototyp AdViz |
| 4 | 20 | 15.5. | CSS, JavaScript allgemein | Abgabe: Prototyp AdViz |
| 5 | 22 | 29.5. | JavaScript: DOM, JSON, AJAX | Aufgabe von Beleg 2: Adviz mit JS |
| 6 | 23 | 5.6. | React Framework | AdViz mit JS |
| 7 | 24 | 12.6. | React Framework | AdViz mit JS |
| 8 | 25 | 19.6. | React Framework | Abgabe: AdViz mit JS |
| 9 | 26 | 26.6. | React Framework / NodeJS | Aufgabe von Beleg 3: AdViz mit Backend |
| 10 | 27 | 3.7. | NodeJS | AdViz mit Backend |

## Heute

- Gestaltung von HTML-Seiten
- Cascading Style Sheets
- Lernziel: CSS in Ihrer adviz-Webapplikation nutzen

## Abgabe von Beleg 1 am 02.05.

- HTML und CSS nutzen für ein “ansprechendes” Design
- Login-Screen und Main-Screen mit Karte sollten den Bildschirm mehrhaltig ausfüllen

## Gestaltung von Webseiten

- “Früher” - Mitte der 1990er:
  - Hierarchische Website-Struktur
  - Eindimensionales Layout, Abschnitte (Paragraphen), Listen
  - HTML-Elemente werden genutzt, um den Inhalt zu formatieren, z.B.,
    - `<body background="images/back.jpg">`
    - `<body bgcolor="#FF0000">`
    - `<CENTER>`
    - `<IMG ALIGN=left src="images/stand1.gif">`
  - Beipspiele:
  - http://oucsace.cs.ohio.edu/~nxie/ (Das ist eine HTML 3.2 Seite!)
  - http://oucsace.cs.ohio.edu/~osterman/ (Das ist eigentlich kein HTML!?)

**Veraltet !**

**Hausaufgabe**: Wenn der Laptop im Netzwerk der HTW registriert ist, bzw. eingeloggt ist, funktioniert diese Seite http://www.ece.ohio.edu/personal/personal.html nicht. Warum?

## Gestaltung von Webseiten

- Ende der 1990er:
  - Ansprechenderes Design,
  - Versuch der Trennung von Struktur des Inhaltes (HTML) und Formatierung des strukturierten Inhaltes (CSS): http://oucsace.cs.ohio.edu/~juedes/ (Das ist eine HTML 4.01 Seite!)
  - Zweidimensionales Layout, angelehnt an das Layout einer Zeitung: http://www.arngren.net/

- Verschiedene Ansätze zum Erstellen dieses 2-dim. Layouts:
  1. Tabellen (gibt es seit Januar 1997) Beispiel: http://toastytech.com/evil/index.html
  2. Frames (gibt es seit Dezember 1997, HTML 4)

     Beispiel: https://docs.oracle.com/javase/1.5.0/docs/api/index.html?overview-summary.html (Das ist eine HTML 4.01 Seite!)
  3. Div-Container & CSS (CSS-Frameworks)

## Gestaltung von Webseiten: Tabellenlayout

- Beispiel: http://toastytech.com/evil/index.html (in Chrome mit Tools  Developer)
- Vorteile:
  - Ansprechenderes Design (gegenüber den “Auflistungs”-Webseiten der Mitt-Neunziger)
  - Genaue Umsetzung von Vorlagen
- Nachteile:
  - Unübersichtlicher HTML-Quellcode
  - **Keine Trennung von Inhalt und Darstellung**
  - Unflexibel bei Änderungen
- Veraltet! Nicht mehr benutzen!

## Gestaltung von Webseiten: Frames

- Idee:
  - Aufteilung in einzelne HTML-Dokumente
  - wiederkehrende Elemente (z.B. Navigation) als eigene HTML-Dokumente
  - Zusammenfügung der Teile in einem zentralen Dokument
  - frameset-Element (statt body), mit frame-Kindern
  - horizontale und vertikale Aufteilung, wenn nötig geschachtelt
- Beispiel:
  - https://docs.oracle.com/javase/1.5.0/docs/api/index.html?overview-summary.html
    (in Chrome mit Tools  Developer)

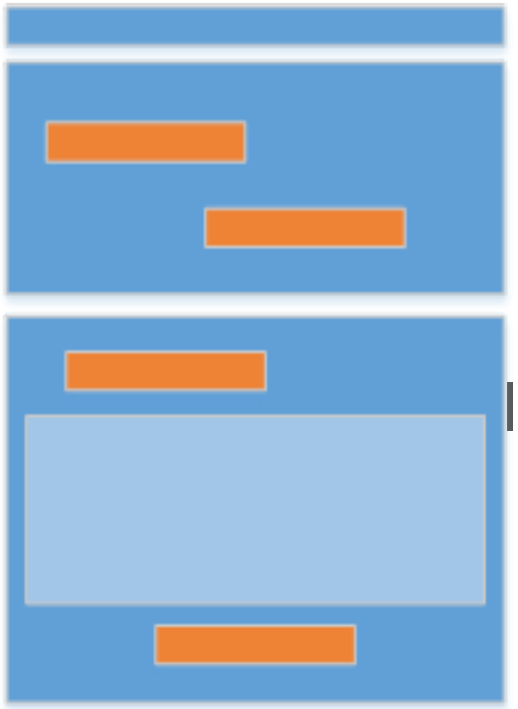
## Gestaltung von Webseiten: Frames

- Vorteile:
  - Wesentlich übersichtlicherer Quellcode als bei Tabellenlayout
  - Sinnvolle Anwendung von Frames: Verzeichnis mit klickbaren Elementen in einem Frame und Frame, wo dann jeweilige Seite angezeigt wird, Nutzer muss nicht zum Verzeichnis zurückspringen
- Nachteile:
  - Nicht alle Browser unterstützen Frames
  - Auf kleinen Bildschirmen zu viele Frames zu unübersichtlich
  - Pro Frame eine HTTP-Verbindung, höhere Ladezeit
- In HTML 5 können <frame>, <frameset> nicht mehr verwendet werden. Es gibt ein <iFrame>-Element zur Einbettung von anderen HTML-Seiten.
- Frames: Veraltet! Nicht mehr benutzen!

## Gestaltung von Webseiten: -s und CSS

Idee:

- Wollen HTML-Element ohne Darstellungseinstellungen:
  - `div` - für Blöcke (volle Breite)
  - `span` – für Elemente im Textfluss
- Unterteilung der Webseiten in `div`-Container
- `div` as in “division”
- Das Layout (Anordnung und Größe) über CSS-In festlegen
- Dadurch Trennung von Inhalt und Darstellung

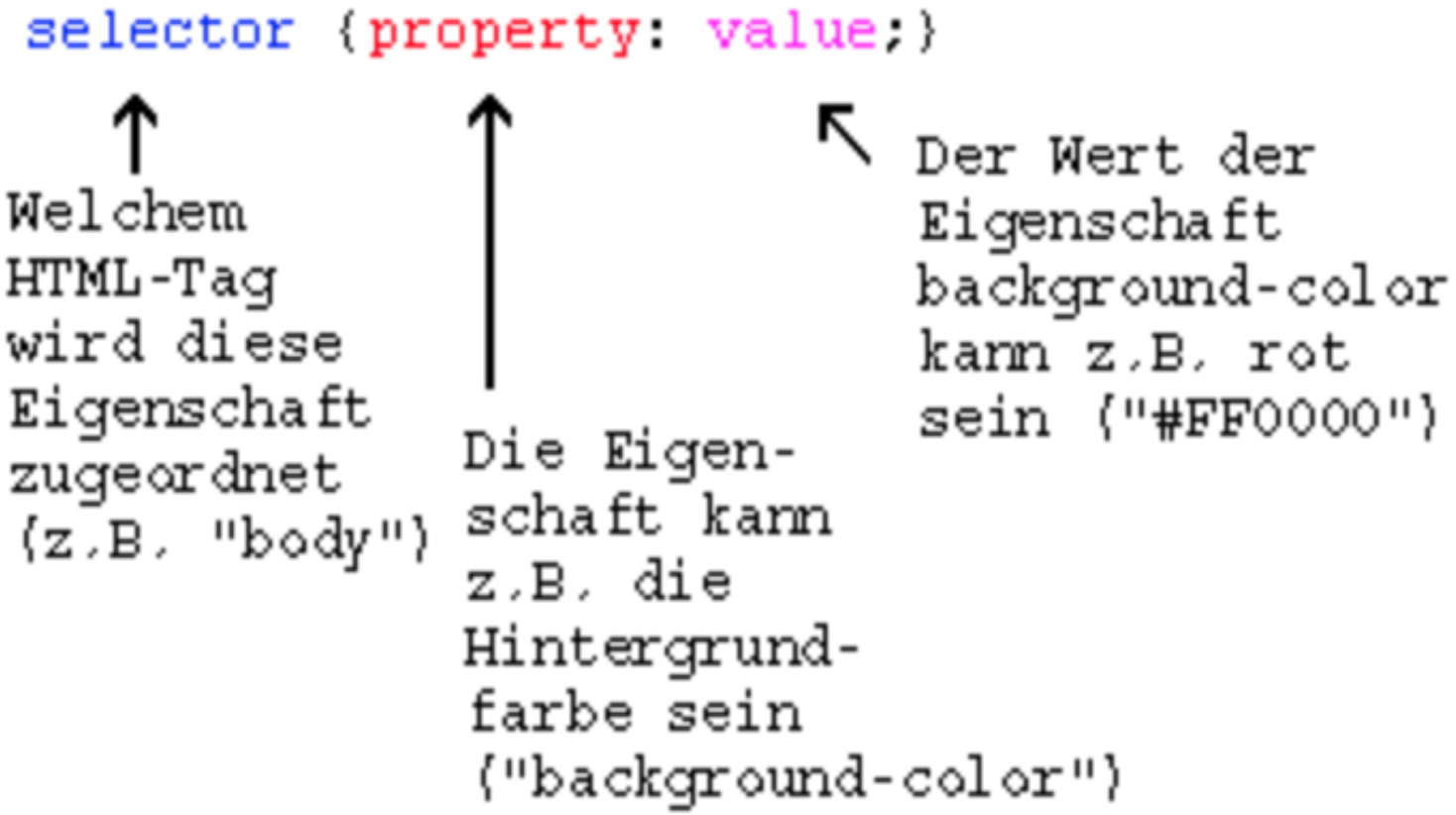


## Was ist denn überhaupt CSS?

- Akronym für **C**ascading **S**tyle **S**heets
- Stilsprache, die das Aussehen von HTML-Elementen definiert
- CSS kann man z.B. zum Festlegen von Schriftarten, Farben, Rändern, Linien, Höhen, Breiten, Hintergrundbildern, für fortgeschrittener Positionierung und viele andere Sachen benutzen.
- Wichtig - Unterschied zwischen HTML und CSS:
  **Mit HTML werden Inhalte strukturiert.**
  **Mit CSS wird der strukturierte Inhalt formatiert.**
- Vorallem: durch diese Trennung ist Pflege der Seiten wesentlich einfacher
- Weitere Vorteile sind u.a.:
  - Kontrolle über das Layout vieler Webseiten aus einer einzigen Style-Sheet-Datei heraus
  - präzisere Kontrolle über das Layout
  - verschiedene Layouts für verschiedene Medientypen (Bildschirmanzeige, Druck etc.);
  - Vielzahl von fortgeschrittenen und anspruchsvolle Techniken, insbesondere mit der letzteren Versionen von CSS
- CSS seit Ende der 1990er Jahre

## Grundlegende CSS-Syntax

- Wollen einen roten Hintergrund unserer Seite:
  - In HTML: `<body bgcolor="#FF0000">` (Veraltet!)
  - Mit CSS: `body {background-color: #FF0000;}`
- Grundlegendes CSS-Model:



## Wo kommt der CSS-Code hin?

- Methode 1: In-line (Attribut-Stil)

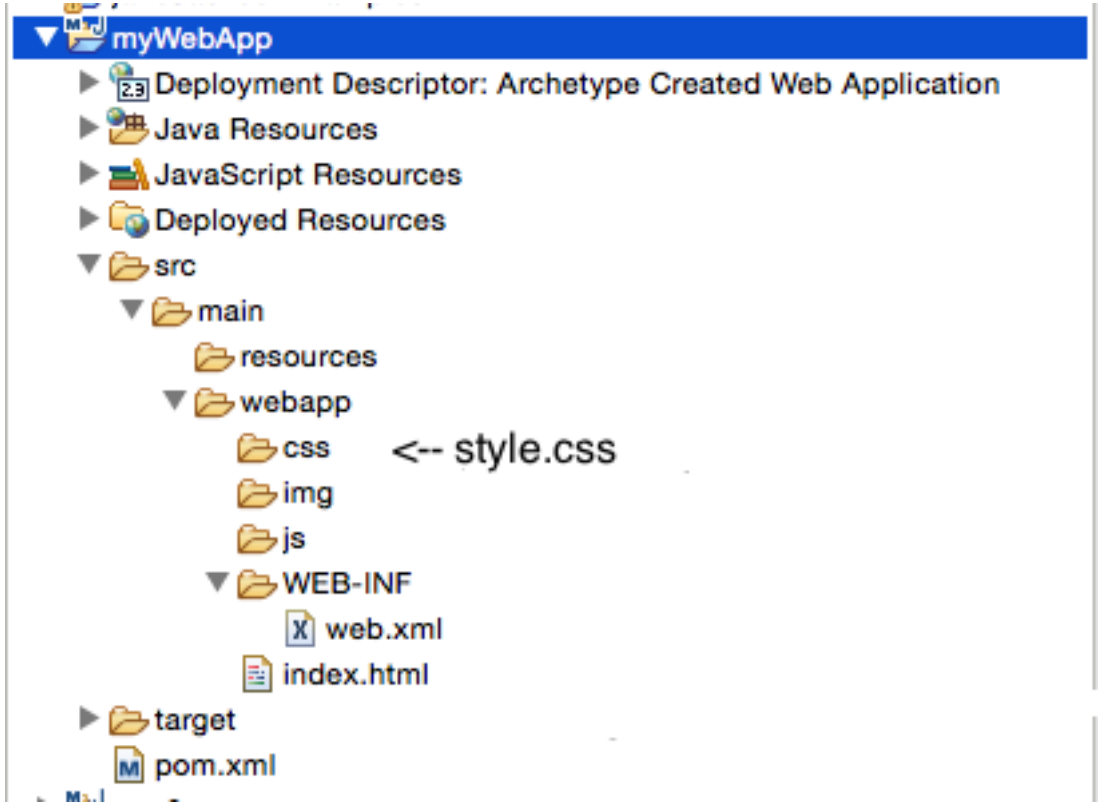
```
<html>
    <head>
        <title>Example</title>
    </head>
    <body style="background-color: #FF0000;">
        <p>Das ist eine rote Seite.</p>
    </body>
</html>
```

- Methode 2: In-document (Tag-Stil)

```
<html>
    <head>
        <title>Example</title>
        <style type="text/css">
            body {background-color: #FF0000;}
        </style>
    </head>
    <body>
        <p>Das ist eine rote Seite.</p>
    </body>
</html>
```

# Wo kommt der CSS-Code hin?

- Methode 3: Verweis auf ein externes Style Sheet
  - Anlegen einer Text-Datei style.css im Verzeichnis src/main/webapp/css




  - Und diese Datei im Kopfteil des HTML-Dokuments referenzieren

## Wo kommt der CSS-Code hin?

- styles.css

```
@charset "UTF-8";
body {
    background-color: #FF0000;
}
```

- index.html:

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/style.css"
/>
    </head>
    <body>
        <h2>Hello World!</h2>
    </body>
</html>
```

## Farben & Hintergründe in CSS

- Können mit den folgenden CSS-Eigenschaften konfiguriert werden:

  color
  background-color
  background-image
  background-repeat
  background-attachment
  background-position
  background

- Beispiel in Eclipse:
  - color of <h2>
  - Wie stelle ich ein Background-Image ein?
- Mehr dazu im Tutorial:

http://de.html.net/tutorials/css/lesson3.php

## Zeichensätze

- Zur Konfiguration von Fonts (Schriftarten) werden diese CSS-Eigenschaften verwendet:
  - font-family
  - font-style
  - font-variant
  - font-weight
  - font-size
  - font

## Zeichensätze: font-family

- Die Eigenschaft ”font-family” wird verwendet, um eine priorisierende Liste der Zeichensätze anzugeben, in der ein bestimmtes Element oder eine Webseite angezeigt werden soll
- Es gibt Schriftarten (fonts), wie:
  - “Arial”
  - “Times New Roman”
  - “Courier”
- Generische Font-Familien, wie:
  - Serif
  - Sans-serif
  - monospace

Times New Roman
Garamond
Georgia

Diese drei Schriftarten gehören zur generischen Familie **serif**. Diese sind an ihren "Füßchen" erkennbar.

Trebuchet
Arial
Verdana

Diese drei Schriftarten gehören zur generischen Familie **sans-serif**. Diese erkennt man daran, dass sie keine "Füßchen" haben.

Courier
Courier New
Andele Mono

Diese drei Schriftarten gehören zur generischen Familie **monospace**. Sie haben alle eine feste Breite.

## Zeichensätze: font-family

- Beispiel für eine priorisierende Liste von Fonts:

```
h1 {
    font-family: arial, verdana, sans-serif;
}
h2 {
    font-family: "Times New Roman", serif;
}
```

- Überschriften, die mit <h1> markiert wurden, werden in der Schrift “Arial” angezeigt. Falls dieser Zeichensatz nicht auf dem Computer des Nutzers installiert ist, wird stattdessen “Verdana” verwendet. Sollten beide Schriften nicht verfügbar sein, wird ein Zeichensatz aus der **sans-serif**-Familie für die Überschriften benutzt.

## Zeichensätze: font-size

- Schriftgröße wird durch die Eigenschaft font-size festgelegt:

```
h1 {
    font-size: 30px; /*absolute Größe */
}
h2 {
    font-size: 12cm; /*absolute Größe */
}
h3 {
    font-size: 120%;
}
p {
    font-size: 1em;
}
```

- **‘%’** und ‘**em**’ erlauben dem Nutzer, die Größe so zu verändern, dass es ihm/ihr passend erscheint. (Beispiel: Browser  View  ZOOM+/-)
- **Um Ihre Webseite für jederman zugänglich zu machen**, sollten Sie veränderbare Einheiten, wie ‘**%**’ oder ‘**em**’, verwenden.

## Zeichensätze: font-size

- Was bedeutet “em”?
- https://developer.mozilla.org/de/docs/Web/CSS/font-size:
  - Die Größe eines em-Wertes ist dynamisch.
  - Beim Spezifizieren der font-size-Eigenschaft entspricht ein em der für das gefragte Elternelement festgelegten Schriftgröße.
  - Wird die Schriftgröße nirgendwo anders auf der Seite festgelegt, dann gilt die vom Browser definierte Größe, die oft 16px entspricht.
  - Demnach gilt im Normalfall 1em = 16px, also 2em = 32px.
  - Wenn dem body-Element eine Schriftgröße von 20px zugewiesen wird, dann gilt 1em = 20px und 2em = 40px. Es ist zu beachten, dass der Wert 2 notwendigerweise ein Multiplikator der aktuellen em-Größe ist.
- Mehr zu Fonts im Tutorial:

http://de.html.net/tutorials/css/lesson4.php

# **Responsive** Schriftgrößen

Responsive Web Design: gestalterisches und technisches Paradigma zur Erstellung von Websites, so dass diese auf Eigenschaften des jeweiligen Endgeräts, vor allem Smartphones und Tabletcomputer, reagieren können.

- px, em, ex, rem und % eignen sich für die Ausgabe auf dem Monitor am besten.
- px war bis zum Erscheinen der mobilen Geräte und hochauflösender Monitore ein exaktes Maß.
- Für **responsive** Webseiten bevorzugen wir heute em und rem.

| Einheit | Bedeutung |
|---|---|
| em | Vertikale Größe der Schrift, relativ zur Schriftgröße des Eltern-Elements. Über den Daumen gepeilt kann 1em als 16 Pixel gerechnet werden. |
| ex | Die x-Höhe der Schrift, relativ zur Schriftgröße des Eltern-Elements |
| rem | font-size wird immer relativ zum HTML-Element berechnet |
| vw | 1 vw = 1% der Breite des Viewports |
| vh | 1 vh = 1% der Höhe des Viewports |
| vmin | 1 vmin ist der kleinere Wert von 1vw und 1vh (der kleineren Seite) |
| vmax | 1 vmax ist der größere Wert von 1vw und 1vh (der größeren Seite) |

- https://www.mediaevent.de/tutorial/masseinheiten.html

## Text

- Zur Formatierung von Texten werden folgende CSS-Eigenschaften verwendet:
  - text-indent
  - text-align
  - text-decoration
  - letter-spacing
  - text-transform
- Mehr dazu im Tutorial:

http://de.html.net/tutorials/css/lesson6.php

## Links

- Darstellungen von Links (<a>-Tags ), z.B. Farbe, Schriftarten, etc. können mit Hilfe von CSS gesteuert werden.
- Link-Darstellung abhängig davon zu gestalten, ob
  - der Link schon besucht wurde (visited)
  - der Link noch nicht besucht wurde (link)
  - der Link  gerade angeklickt wird (active)
  - der Mauszeiger über dem Link platziert ist (hover)
- Dazu werden Pseudo-Klassen benutzt

```
a:link {  /* der noch nicht besucht Link ist blau und nicht unterstrichen */
    color: blue;
    text-decoration:none;
}
a:visited { color: red; } /* der besuchte Link ist rot*/

a:active { background-color: #FFFF00; } /* beim Clicken wird der Hintergrund
gelb*/

a:hover { font-style: italic; } /* bei Hover wird der Link-Font italic*/
```

## class und id

- Es geht um die Zuweisung eines speziellen Stil zu einer Gruppe von Elementen oder zu einem individuellen Element
- Beispiel: wein.html

```
<p>Grapes for white wine:</p>
<ul>
    <li><a href="ri.htm">Riesling</a></li>
        <li><a href="ch.htm">Chardonnay</a></li>
    <li><a href="pb.htm">Pinot Blanc</a></li>
</ul>
<p>Grapes for red wine:</p>
<ul>
    <li><a href="cs.htm">Cabernet Sauvignon</a></li>
    <li><a href="me.htm">Merlot</a></li>
    <li><a href="pn.htm">Pinot Noir</a></li>
</ul>
```

**Trauben für Weißwein:**

- Riesling
- Chardonnay
- Pinot Blanc

**Trauben für Rotwein:**

- Cabernet Sauvignon
- Merlot
- Pinot Noir

- Wollen: Weißweinlink sollen orange werden, Rotweinlinks sollen braun werden

## class

```
<p>Grapes for white wine:</p>
<ul>
    <li><a href="ri.htm" class="weisswein" >Riesling</a></li>
        <li><a href="ch.htm" class="weisswein" >Chardonnay</a></
li>
    <li><a href="pb.htm" class="weisswein" >Pinot Blanc</a></li>
</ul>
<p>Grapes for red wine:</p>
<ul>
    <li><a href="cs.htm" class="rotwein" >Cabernet
Sauvignon</a></li>
    <li><a href="me.htm" class="rotwein" >Merlot</a></li>
    <li><a href="pn.htm" class="rotwein" >Pinot Noir</a></li>
</ul>
```

```
a {  color: blue; }
a.weisswein { color: #FFBB00; }
a.rotwein { color: #800000; }
```

Das ganze sieht dann so aus:

http://de.html.net/tutorials/css/lesson7_ex2.php

## id - Identifikation eines bestimmten Elementes

- Nur ein spezielles Element soll anderes dargestellt werden
- Nur die Überschrift Kapitel 1.2 soll rot werden:

```
<h1 id="k1">Kapitel 1</h1>
...
<h2 id="k1-1">Kapitel 1.1</h2>
...
<h2 id="k1-2">Kapitel 1.2</h2>
...
<h1 id="k2">Kapitel 2</h1>
...
<h2 id="k2-1">Kapitel 2.1</h2>
...
<h3 id="k2-1-2">Kapitel 2.1.2</h3>
...
```

```
#k1-2 {
    color: red;
}
```

- Das sieht dann so aus:
  http://de.html.net/tutorials/css/lesson7_ex3.php

##  und 

- werden zum Gruppieren und Strukturieren eines HTML Dokumentes benutzt
- oft mit Attributen “class” und “id”
- haben eine **zentrale** Bedeutung für den Aufbau von Webseiten mit Hilfe von CSS ist
- Gruppieren mit : wird für Gruppierung von einem oder mehreren Block-Level-Elementen (<p>, <ul>, <ol>) verwendet
- Gruppieren mit : wird für Hervorhebungen im Textfluss benutzt (inline), z.B., innerhalb eines Block-Level-Elements

## 

-  tut nix, ist ein neutrales Element
- In Verwendung mit dem Attribut “class”  sehr nützlich
- Im folgenden HTML sollen die Wörter ”Edel”, “hilfreich”, “gut” in rot geschrieben werden

```
<p>Edel sei der Mensch, hilfreich und gut.</
p>
```

```
<p>
<span class="tugend">Edel</span> sei der Mensch,
<span class="tugend">hilfreich</span> und
<span class="tugend">gut</span>.
</p>
```

```
span.tugend {
color:red;
}
```

- So sieht das dann aus: http://de.html.net/tutorials/css/lesson8_ex1.php

## 

```
<div id="spd">
    <ul>
    <li>Gerhard Schröder</li>
    <li>Helmut Schmidt</li>
    <li>Willy Brandt</li>
    </ul>
</div>
 <div id="cducsu">
    <ul>
    <li>Angela Merkel</li>
    <li>Helmut Kohl</li>
    <li>Kurt Georg Kiesinger</li>
    </ul>
 </div>
```

```
#spd {
    background:red;
    color:white;
}
#cducsu {
    background:black;
    color:white;
}
```

# Aufbau von CSS

```
Selektor [, Selektor2, ... ] {
      Eigenschaft: Wert;
      Eigenschaft: Wert1 Wert2...;
      /* ... */
      Eigenschaft-xxx: Wert-xxx;
}
```

```
body {
      max-width: 600px;
      margin: 0 auto;
      font-size: 18px;
      line-height: 1.4;
}
```

- Selektoren:

```
* { /* Sämtliche HTML-Elemente */ }
h1 { /* Alle <h1>-Überschriften */ }
.logo { /* Alle HTML-Elemente mit der Klasse "logo" */ }
#htwlogo { /* Das HTML-Element mit der Id "htwlogo" */ }
img[alt] { /* Alle Bilder mit einem alt-Attribut */ }
```

## Aufbau von CSS (Kombinatoren)

```
/* Nachfahren – alle Links im <nav>-Tag */
/* Hinweis: <nav> (Navigation) ist eine Liste von Links */
nav a { }

/* Kinder – alle direkten Nachfahren von <nav> */
nav > a { }

/* Nachbar – das <p>-Tag, das direkt unter </h1> folgt */
h1 + p { }

/* Geschwister – alle <p>-Tags, die auf gleicher Ebene auf <h1>
folgen */
h1 ~ p { }
```

CSS-Selektoren in der Übersicht:

https://wiki.selfhtml.org/wiki/CSS/Selektoren

## Die Kaskade

- Mehrere Selektoren können für ein Element gelten, Beispiel:

```
HTML:
<p class="green"><a class="green"
id="red">Link</a></p>

CSS:
body { font-size: 200%; }
#red { color: red; }              /* 1-0-0 */
.green { color: green; }              /* 0-1-0 */
p a { color: orange; }            /* 0-0-2 */
a { color: blue; }                /* 0-0-1 */
```

- 4 Selektoren gelten für das <a>-Element
- Was nun?

## Die Kaskade

- Herkunft des Stylesheets:
  1. **Browser-Stylesheet**: Browser besitzen vorgegebene Formatierungen für HTML-Dokumente (geringe Priorität)
  2. **Benutzer-Stylesheet**: Benutzer können in den Einstellungen ihres Browser bevorzugte Schriftarten oder Farbmischungen festlegen. Benutzer-Stylesheets überschreiben Eigenschaften im Browser-Stylesheet (mittlere Priorität und überschreibt das Browser-Stylesheet)
  3. **Autoren-Stylesheet**: Diese sind vom Dokument selbst eingebunden oder mit dem Dokument mitgeliefert. **Autoren-Stylesheets überschreiben Browser- und Benutzer-Stylesheets.**

- Das Browser-Stylesheet bildet die Basis und besitzt geringe Priorität. Das Benutzer-Stylesheet besitzt mittlere Priorität und überschreibt das Browser-Stylesheet. Autoren-Stylesheets besitzen die höchste Priorität und überschreiben sowohl Browser- als auch Benutzer-Stylesheets.

## Die Kaskade

- Prüfung pro Element anhand der Selektoren:
  - keine Deklaration → Erben von Vorfahren im DOM
  - genau eine Deklaration → Verwenden
  - mehrere Deklarationen → Kaskade
    1. nach Wichtigkeit: Deklarationen mit `!important` und Ursprung des Stylesheets
    2. nach Spezifität: Elementname < Klasse < ID < Inline –Regeln sind am spezifischsten
    3. nach Reihenfolge: zuletzt deklarierte Eigenschaft wird verwendet, Beispiel:

```
h1 { color: red; }
h1 { color: orange; color: green; }
```

h1-Überschrift ist grün

- http://wiki.selfhtml.org/wiki/CSS/Kaskade
- https://little-boxes.de/artikelansicht/die-kaskade-im-schnelldurchgang.html

## Die Kaskade

- Mehrere Selektoren können für ein Element gelten, Beispiel:

```
HTML:
<p class="green"><a class="green"
id="red">Link</a></p>

CSS:
body { font-size: 200%; }
#red { color: red; }              /* 1-0-0 */
.green { color: green; }              /* 0-1-0 */
p a { color: orange; }          /* 0-0-2 */
a { color: blue; }                /* 0-0-1 */
```

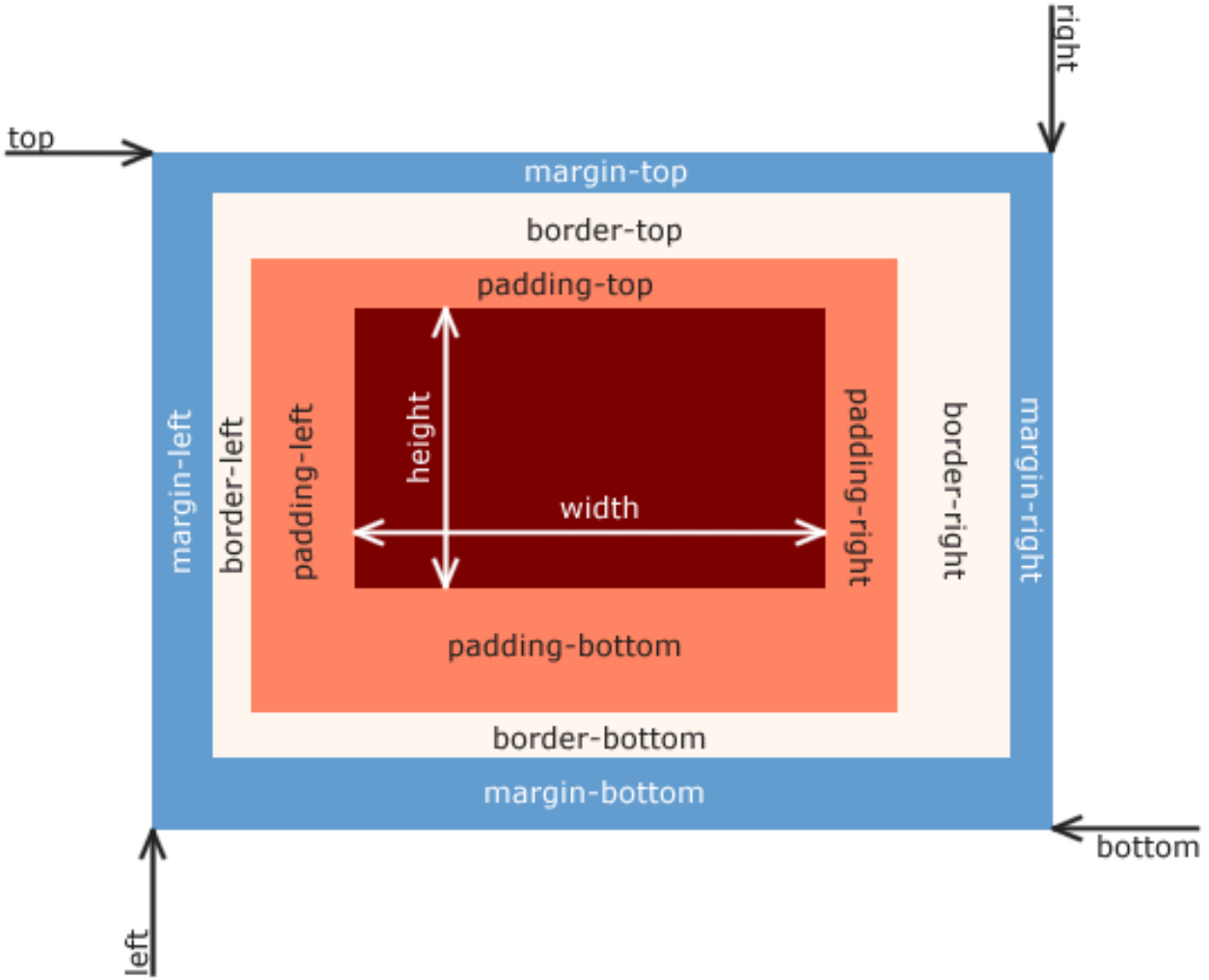
- 4 Selektoren gelten für das <a>-Element:

1. Zwei Elementnamen schlagen einen Elementnamen: 0-0-2
2. Eine Klasse schlägt Elementnamen (egal wie viele): 0-1-0
3. Eine ID schlägt wiederum Klassennamen: 1-0-0

Link

## Boxmodell

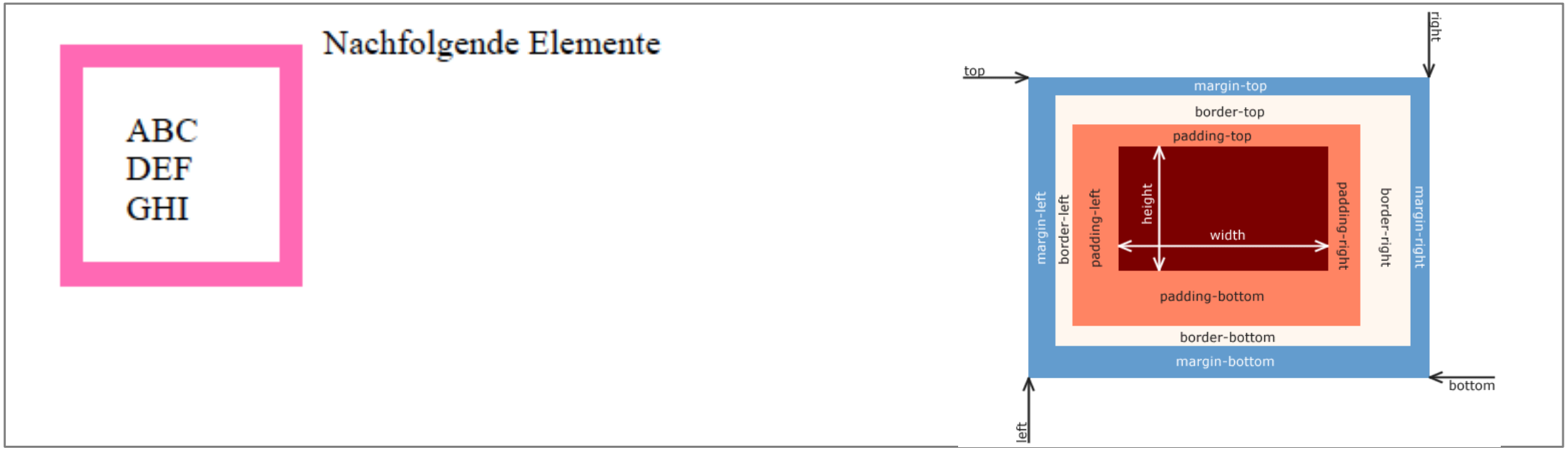
- **Jedes HTML-Element** nimmt einen gewissen Raum auf der Webseite ein, der detailliert festgelegt werden kann:




Source: https://upload.wikimedia.org/wikipedia/commons/7/7a/Boxmodell-detail.png

# Boxmodell

```
HTML:
<div class="custom-box">ABC DEF GHI</div>Nachfolgende Elemente
CSS:
.custom-box {
      width: 50px;
      height: 50px;
      margin: 10px;
      padding: 20px;
      border: 10px solid hotpink;
}
```



## Positionierung von HTML-Elementen

`position` (Eigenschaft)

Werte:

- `static` - Default ( `top`,`bottom`,`left`,`right` werden ignoriert)
- `relative` - Bleibt in normalem Fluss und wird von den anderen Elementen verschoben, Bezugspunkt für absolute positionierte Kindelemente
- `absolute` - Relativ zum nächsten Vorfahren (mit position != static )
- `fixed` - Relative Position zum Viewport des Browsers (losgelöst vom Textfluss)
- `sticky` - Bleibt am oberen oder unteren Seitenrand "kleben" (nicht in allen Browsers)
- Korrespondierende CSS-Eigenschaften sind: top, bottom, left, right

Mehr dazu:
https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Positionierung/position

# Positionierung von HTML-Elementen

- IDE: index.html und style.css, in chrome angucken

CSS:

```
.abs {
      position: relative;
      padding: 4em;
      background-color: rgba(0, 130, 209, .3);
}
.o-r { top: 0; right: 0 }
.u-l { bottom: 0; left: 0 }
```

HTML:

```
<div class="abs o-r u-l">
Überall
<span class="abs o-r"> oben rechts</span>
<span class="abs u-l"> unten links</span>
</div>
```

# CSS Advanced

- Tutorial on CSS:

https://wiki.selfhtml.org/wiki/CSS/Tutorials/Einstieg

- CSS Advanced:
  - Text Effects
  - Tooltips
  - Shadows, Gradients
  - Transformations
  - Animations
  - Transitions
  - ...

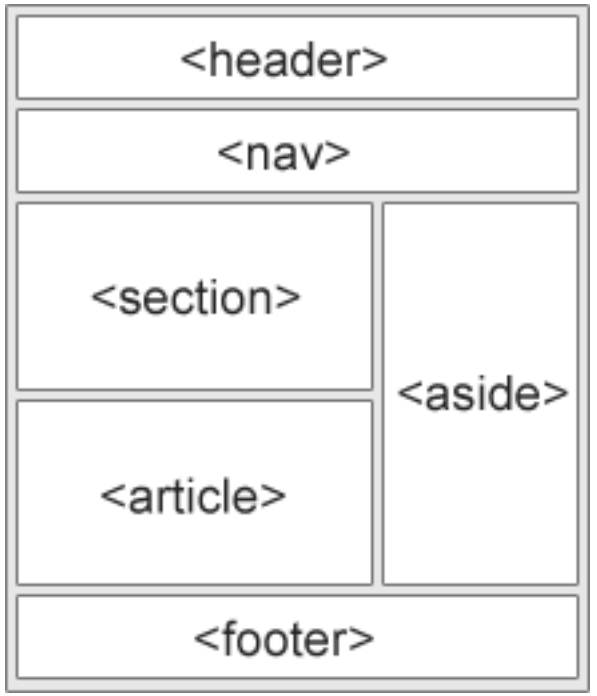
- CSS-Functions:

https://wiki.selfhtml.org/wiki/CSS/Wertetypen/Funktionen

## Layout

- In HTML 5 gibt es neue semantische/strukturelle Elemente
- Diese Elemente definieren verschiedene Teile einer Webseite:

| Tag | Description |
|---|---|
| <article> | Defines an article in a document |
| <aside> | Defines content aside from the page content |
| <figcaption> | Defines a caption for a <figure> element |
| <figure> | Defines self-contained content |
| <footer> | Defines a footer for a document or section |
| <header> | Defines a header for a document or section |
| <main> | Defines the main content of a document |
| <nav> | Defines navigation links |
| <section> | Defines a section in a document |

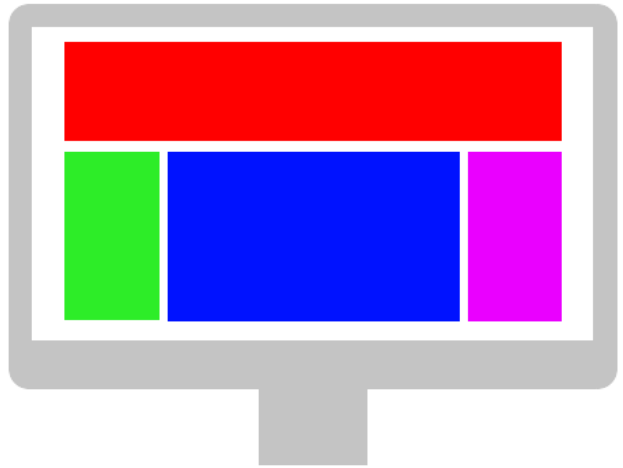



Vollständige Tabelle und eine Liste für alle HTML 5 Neuheiten hier:
https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

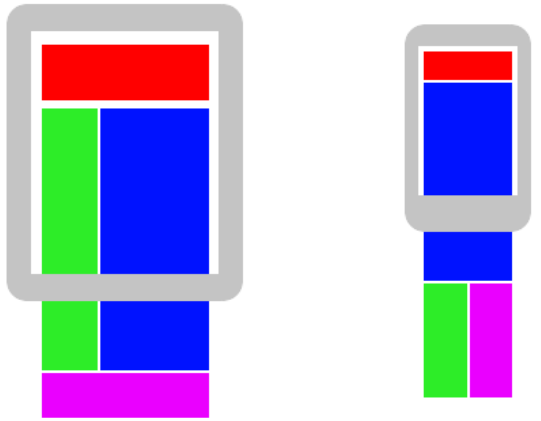
# CSS Layouts

- Responsive Web Design:

gestalterisches und technisches Paradigma
Erstellung von Websites, so dass diese auf Eigen
jeweiligen Endgeräts, vor allem Smartph
und Tabletcomputer, reagieren können



- Verschiedene “responsive” CSS Layout Varianten
- Zum Beispiel: das Grid Layout
  - seit 2017
  - das Mittel der Wahl, um …
  - Inhalte in responsive Raster (Grids) zu fassen
  - Raster passen sich dem Viewport automatisch a



Bei Interesse, Tutorial angucken:
https://developer.mozilla.org/de/docs/Web/CSS/CSS_Grid_Layout

# CSS Layout Templates

https://wiki.selfhtml.org/wiki/CSS/fertige_Layouts

# The End

- Questions?
- References:
  - HTML5 Tutorial: https://wiki.selfhtml.org/wiki/HTML/Tutorials/HTML5/Grundger%C3%BCst
  - CSS-Tutorial auf Deutsch: https://developer.mozilla.org/de/docs/Learn/Getting_started_with_the_web/CSS_basics
  - Dane Cameron: “A Software Engineer Learns HTML5, JavaScript & jQuery”, Cisdal Publishings, 2015
  - Frank Zieris, Webentwicklung, WiSe2017/18, HTW: https://www.zieris.net/teaching/htw-berlin/webdev-slides/?p=12_websites