CSS – Cascading Stylesheets

CSS – Einführung

- CSS: **C**ascading **S**tyle **S**heets
- Bestimmt die Darstellung von HTML-Elementen
- CSS löste ein Problem
  - HTML war ursprünglich nicht dafür entwickelt worden, Formatierungen zu beinhalten
  - HTML 3.2 führte Formatierungstags ein, welche die Entwicklung jedoch kompliziert machten (z.B. <i>, <b> für kursive und fette Schrift)
  - Änderungen mussten in jedem HTML-Dokument einzeln vorgenommen werden
  - Als Lösung für das Problem führte das W3C CSS ein
- CSS trennt Design vom Inhalt
  - Alle Formatierungen sollten in eine gesonderte `.css` Datei ausgelagert werden
  - So lässt sich die Darstellung der Website in nur einer Datei bearbeiten

CSS – Versionen

- CSS 1 (1996)
  - Erste Version von CSS
  - Unterstützt Fonts, Farben, Alignments, etc.
  - Von allen aktuellen Browsern unterstützt
- CSS 2 (1998)
  - Führte u.a. absolute und relative Positionen ein
  - War zunächst fehlerhaft und daher wenig beliebt, heutzutage vom W3C nicht mehr empfohlen
- CSS 2.1 (2011)
  - Überarbeitete Version von CSS 2, behob einige Fehler und Unstimmigkeiten
  - Mittlerweile von den meisten gängigen Browsern korrekt implementiert
- CSS 3
  - Seit 2000 in Entwicklung
  - CSS Namespaces, Media Queries, u.v.m.

CSS – Syntax

```
h1 {color:blue; font-size:12px;}
```

Selektor

Deklaration
Eigenschaft :Wert

Deklaration
Eigenschaft: Wert

- Selektor wählt das betroffene HTML-Element aus
- Deklarationsblock enthält die Formatierungen
- Formatierung wird für alle passenden Elemente übernommen

```
<html>
…
    <h1> Dies ist eine Überschrift </h1>
…
</html>
```

Dies ist eine Überschrift

CSS – Einbindung

Externe Einbindung

- Empfohlen für CSS-Regeln, die mehr als nur eine Seite betreffen
- Textdatei mit Dateiendung `.css`
- Einbindung in das HTML-Dokument über Link

```
<head>
  <link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Interne Einbindung

- Sollte nur verwendet werden, wenn die CSS-Regeln nicht mehr als eine Seite betreffen
- CSS-Regeln werden innerhalb von `<style>`-Tags definiert

```
<head>
  <style>
    <!-- CSS-Regeln -->
  </style>
</head>
```

CSS – Selektoren

Einleitung

- Selektoren
  - Dienen dazu, die zu formatierenden Elemente des HTML-Dokuments auszuwählen
  - Dürfen nur aus Groß- oder Kleinbuchstaben (a-z, A-Z), Ziffern (0-9) und dem Bindestrich (-) bestehen und müssen mit einem Buchstaben beginnen
- Aufbau einer CSS-Anweisung
  - `Selektor, Selektor … { Anweisung:Wert; Anweisung:Wert; … }`
- 4 Typen von Selektoren

Format	Selektor	Selektierte Elemente	Beispiel
`span`	Element	Selektieren alle **span**-Elemente im Dokument	`span {...}`
`.Klasse1`	Klasse	Selektieren alle Elemente mit der Klasse „Klasse1“ im **class**-Attribut	`.Klasse1 {...}`
`#logo`	ID	Selektieren genau das Element mit dem **id**-Attribut-Wert „logoWrapper“	`#Logo {...}`
*	Universal	Selektiert alle Elemente des Dokuments	`* {...}`

CSS – Selektoren

Verschachtelung – An Beispielen

- Typ-Selektoren können ein Element präzisieren
  - Selektiert alle **span**-Elemente mit der Klasse „Klasse1“

```
span.Klasse1 {...}
```

  - Selektiert nur **span**-Elemente mit der ID „logo“

```
span#logo {...}
```

  - Selektiert alle **span**-Elemente mit der Klasse „wrapper“ und der ID „logoWrapper“

```
span.wrapper#logoWrapper {...}
```

- Typ-Selektoren können geschachtelt werden
  - Nachfahren-Selektor

```
p span {...}
```

    - Alle **span**-Elemente, die sich innerhalb von Paragraphen befinden, z.B.:

```
<p>
    <span>Selektiert</span>
    <div><span>Selektiert</span></div>
</p>
```

  - Kind-Selektor

```
p>span {...}
```

    - Alle direkten **span**-Element-Kinder von Paragraphen, z.B.:

```
<p>
    <span>Selektiert</span>
    <div><span>Nicht selektiert</span></div>
</p>
```

CSS – Selektoren

Verschachtelung – An Beispielen

- Typ-Selektoren können geschachtelt werden (Forts.)
  - Nachbar-Selektor

```
div + p {...}
```

```
<div></div>
<p>Selektiert</p>
<p>Nicht Selektiert</p>
```

  - Attribut-Selektor

Format	Erklärung	Beispiel
`Sel.[Attr]`	Das Attribut muss vorhanden sein	`img[alt] {...}`
`Sel.[Attr="Wert"]`	Das Attribut muss den Wert „Wert“ haben	`input[type="checkbox"] {...}`
`Sel.[Attr^="Wert"]`	Der Wert des Attributs muss mit „Wert“ beginnen	`a[href^="http:"] {...}`

CSS – Selektoren

Pseudoklassen

- Pseudoklassen werden durch Benutzerinteraktion oder durch die Struktur des HTML-Dokuments vergeben, z.B.:
  - Überfahren eines Links → :hover
  - Erstes Element in einer geordneten Liste → :first-child

Format	Erklärung	Beispiel
`Selektor:hover`	Überfahren eines Links	`a:hover{...}`
`Selektor:active`	Link ist angeklickt	`a:active{...}`
`Selektor:visited`	Besuchter Link	`a:visited{...}`
`Selektor:link`	Unbesuchte Links	`a:link{...}`
`Selektor:focus`	Element ist fokussiert	`input[type="text"]:focus{...}`
`Selektor:first-child`	Erstes Kind des Elements	`#linkliste:first-child{...}`
`Selektor:not(S)`	Selektiert alle Elemente, die den Selektor S nicht besitzen	`img:not(.thumbnail){...}`

- Weitere Selektoren siehe: http://w3schools.com/cssref/css_selectors.asp

CSS – Hintergründe

- Hintergrundfarbe
  - Mit `background-color:lightgrey` wird die Hintergrundfarbe festgelegt (hier: hellgrau)
- Hintergrundbild
  - Mit `background-image:url("logo.png")` wird die URL zur Bilddatei angegeben
- Positionierung
  - `Background-position: bottom right` gibt die Position des Bildes an
  - Der Rest der Seite bekommt die festgelegte Hintergrundfarbe (hellgrau)
  - `bottom, top, center, left, right` können alleine oder auch in Kombination verwendet werden
  - Alternativ kann der horizontale und vertikale Abstand zum Bildschirmrand angegeben werden
    Beispiel: `background-position: 50% 30px;`

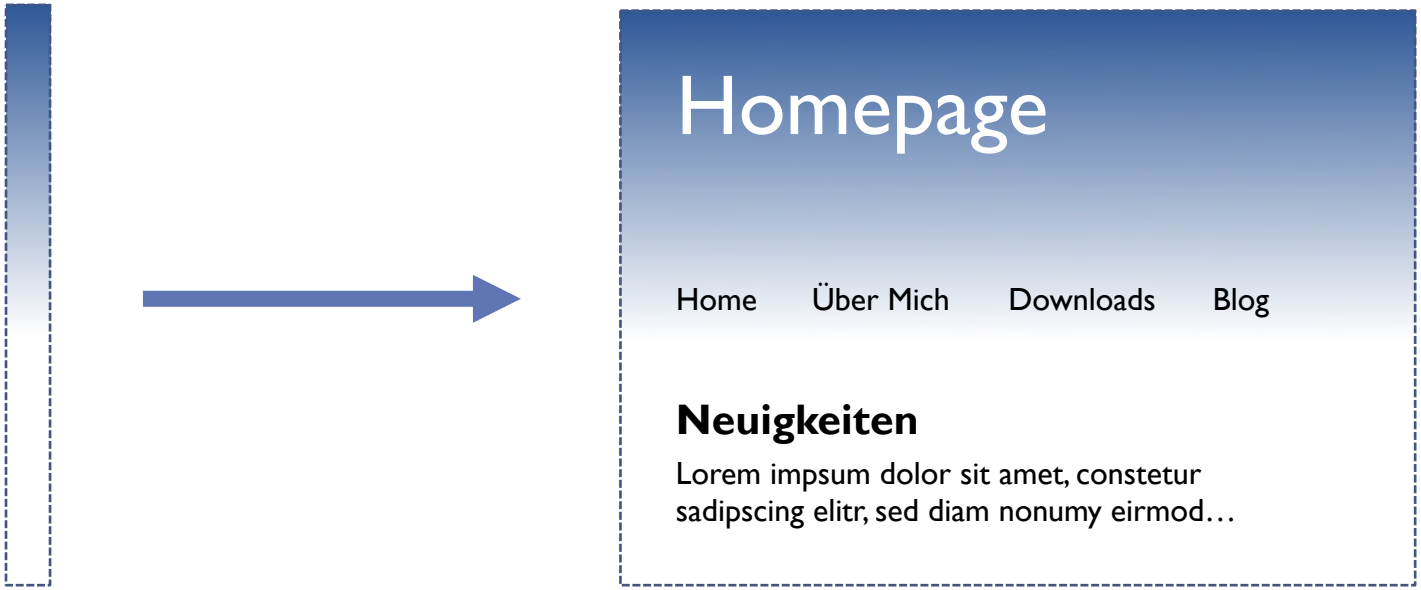
CSS – Hintergründe

- Hintergrundbild – Wiederholung horizontal oder vertikal
  - `background-repeat: no-repeat` gibt an, wie oft das Bild wiederholt werden soll
  - `no-repeat:` keine Wiederholung
  - `repeat:` das Bild wird horizontal und vertikal wiederholt
  - `repeat-x:` das Bild wird nur horizontal wiederholt
  - `repeat-y:` das Bild wird nur vertikal wiederholt

- Background-attachment
  - `background-attachment: fixed` gibt an, ob sich das Hintergrundbild beim Scrollen mitbewegt
  - `fixed:` Bild scrollt nicht mit
  - `scroll:` Bild scrollt mit

CSS – Hintergründe

- Beispiel für horizontale Wiederholung
  - Kleines Ausgangsbild spart Bandbreite und erhöht die Ladegeschwindigkeit
  - Hintergrund beliebig skalierbar



CSS – Hintergründe

Mehrere Hintergrundbilder

- Mehrere Hintergrundbilder für ein Element
  - Mit `background-image` werden die URLs der Bilder komma-separiert angegeben
  - Die Reihenfolge bei `background-image` ist ausschlaggebend

```
background-image: url(logo.png), url(verlauf.png);
background-position: bottom right, 100% center;
background-repeat: no-repeat, repeat-x;
```

  - Der jeweils erste Wert der Attribute bezieht sich auf das erste Bild (logo.png) und der zweite Wert auf das zweite Bild (verlauf.png)
  - Durch mehrere Hintergrundbilder kann man zum Beispiel Hintergründe dynamisch an die Bildschirmgröße anpassen (Responsive Design)

CSS – Hintergründe
Mehrere Hintergrundbilder

- Beispiel

```
background-image: url(logo.png), url(verlauf.png);
background-position: bottom right, top left;
background-repeat: no-repeat, repeat-x;
```

CSS – Größeneinheiten

Einheit	Verhältnis	Beschreibung
in (Inch)	1 in = 2.54 cm	Größenangabe in Inch
cm (Zentimeter)	1 cm = 10 mm	Größenangabe in Millimetern
mm (Millimeter)	10 mm = 1 cm	Größenangabe in Zentimetern
pt (Punkte)	1 pt = 1/72 in = 0,3527... mm	Größenangabe in (Bild-)Punkten (Vereinheitlicht durch 1/72 Inch definiert)
pc (Pica)	1 pc = 12 pt	Größenangabe in Pica
em (Schriftgröße)	1 em = 100% Schriftgröße	Größenangabe in Prozent, relativ zur Standard-schriftgröße
ex (x-Schriftgröße)	1 ex = 100% Schriftgröße des kleinen x	Größenangabe in Prozent, relativ zum kleinen x in Standardgröße
px (Pixel)	1 px = 1 Pixel	Größenangabe relativ zur Auflösung 1 px entspricht einem Pixel des Bildschirms

Quelle: http://www.html-world.de

CSS – Fonts

- Schriftarten auswählen
  - Mit `font-family` lassen sich Schriftarten auswählen
  - Achtung: Schriftart muss auf dem System des Benutzers installiert sein
  - Es sollten stets mehrere Schriftarten angegeben werden, falls der Browser die gewünschte nicht unterstützt
  - Beispiel: `font-family: "Times New Roman", Times, serif`
  - Erster Parameter: die bevorzugte Schriftart
  - Letzter Parameter: generische Font-Family, damit der Browser selbst eine ähnliche Schriftart wählen kann

CSS – Fonts

- `font-style` bestimmt den Schrifttyp
  - `normal` für gewöhnlichen Text
  - `italic` für kursive Schrift

- `font-weight` bestimmt die Dicke der Buchstaben
  - `normal` für gewöhnliche Buchstaben
  - `bold` für fette Buchstaben
  - `100, 200 ,…, 900` für genau bestimmte Dicke
    - `400` ist wie `normal`, `700` ist wie `bold`

CSS – Fonts

Webfonts

- Benötigen keine Installation im Browser
  - Werden automatisch von einer Online-Ressource bezogen
  - Einbindung mit `@font-face`

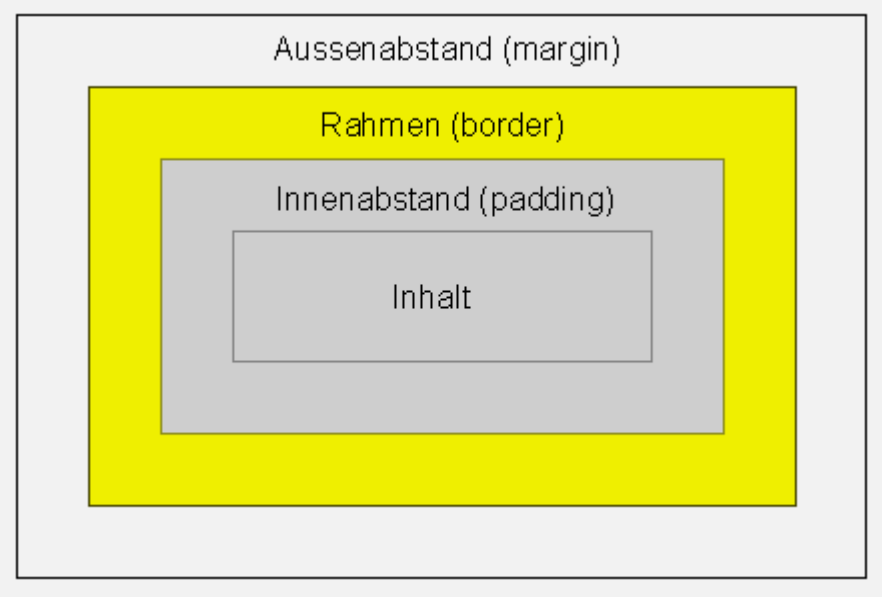
```
@font-face{
      font-family: 'Calligraffitti' ;
      font-style: normal;
      font-weight: normal;
      src: local('Calligraffitti'),
                  url('http://... /... /calligraffitti.otf');
 }
```

  - Google bietet eine Reihe von frei verwendbaren Schriftarten an
    - https://fonts.google.com/
  - Externe CSS-Dateien mit den Schriftart-Definitionen werden, wie gewohnt, mit dem `<link>`-Tag eingebunden

```
<link href = "http://fonts.googleapis.com/css?family=Calligraffitti"
      rel="stylesheet" type="text/css">
```

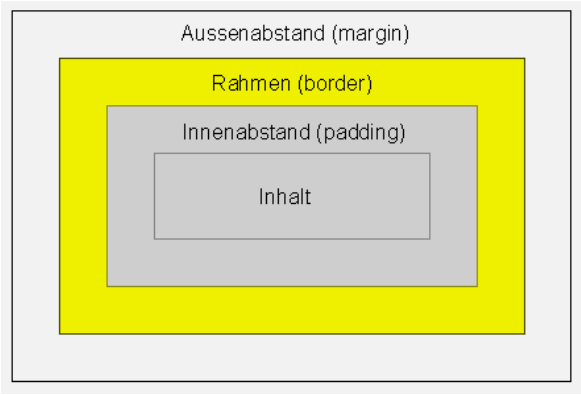
CSS – Boxmodell

- Jedes HTML Blocklevel Element ist von einem Rahmen umgeben
- Mit `margin`, `padding` und `border` gibt man den Außenabstand, Innenabstand und die Rahmenbreite an



CSS – Boxmodell

- Border
  - `border-width` gibt die Rahmenbreite an
    - Längenangaben: z.B. `em, ex, px`
    - Vordefiniert: `thin, medium, thick`
  - `border-color` gibt die Farbe an
  - `border-style` gibt die Form an
    - `None, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset`



  - Will man nur eine bestimmte Seite des Rahmens adressieren, kann man dies mit entsprechenden Attributen tun
  - z.B. `border-left-style` für nur die linke Seite des Rahmens

CSS – Textschatten

- Das Attribut `text-shadow` erlaubt es, Texte mit Schatteneffekten zu versehen

- Beispiel: `text-shadow: -4px 4px 2px dimgrey`
  - Der erste Wert (-4px) gibt die **horizontale** Verschiebung an
    Negative Werte versetzen den Schatten nach Links und positive Werte nach rechts
  - Der zweite Wert (4px) gibt die **vertikale** Verschiebung an
    Negative Werte versetzen den Schatten nach oben und positive Werte nach unten
  - Der dritte Wert (2px) gibt die **Schärfe** des Schattens an
    Je größer der Wert, umso mehr wird der Schatten verwischt (0px = scharfe Kanten)
  - Der letzte Wert gibt die **Farbe** des Schattens an

CSS – Textschatten

- Beispiel

```
<!DOCTYPE html>
<html>
<head>
<style>
    h1 {text-shadow: -4px 4px 2px dimgrey;}
</style>
</head>
<body>
    <h1>Text mit Schatteneffekt!</h1>
</body>
</html>
```

CSS – Box Schatten

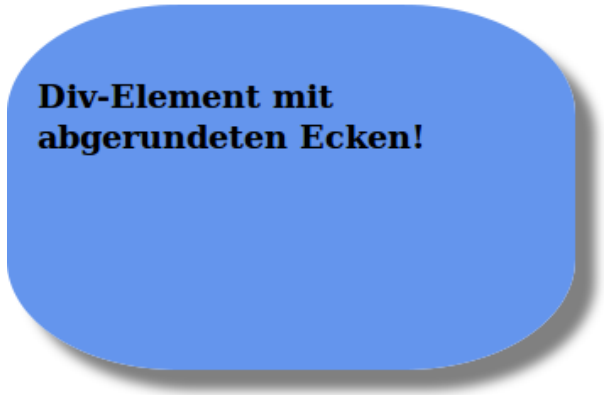
- Das Attribut `box-shadow` verleiht jedem Element einen Schatteneffekt
  - Selbes Prinzip wie bei `text-shadow`

```
<head>
<style>
    div {
    box-shadow: 10px 10px 5px grey;
    width: 300px;
    height: 70px;
    padding: 15px;
    background-color: cornflowerblue;
    font-weight: bold;
}
</style>
</head>
<body>
    <div> <br> Div-Element mit Schatteneffekt!
    </div>
</body>
```

CSS – Abgerundete Ecken

- Das Attribut `border-radius` erlaubt es, HTML-Elemente mit abgerundeten Ecken zu versehen
- Beispiel: `border-radius: 30%`
  - Je höher der Wert, umso runder die Ecken
  - Bei einem Wert ab 50% sind keine geraden Kanten mehr vorhanden

```
<head> <style>
div {
    border-radius: 30%;
    box-shadow: 10px 10px 5px grey;
    width: 250px;
    height:150px;
    padding: 15px;
    background-color: cornflowerblue;
    font-weight: bold;
}
</style> </head>
<body>
<div> Div-Element mit abgerundeten Ecken! </div>
</body>
```



CSS – Farben

- CSS bietet unterschiedliche Techniken zur Farbwahl
  - Mit unterschiedlichen Techniken können dieselben Farben erzielt werden
  - Keine Technik ist der anderen überlegen
  - Wahl der Technik abhängig von Gewohnheit oder persönlicher Präferenz

- RGBA(Rot, Grün, Blau, Alpha)
  - Die Werte rot, grün und blau können zwischen 0 und 255 liegen
  - Alpha beeinflusst die Transparenz der Farbe
  - Der Wert alpha kann zwischen 0.0 (völlige Transparenz) und 1.0 (Blickdicht) liegen
  - Wenn keine Transparenz gewünscht ist, kann statt dessen auch RGB(Rot, Grün, Blau) verwendet werden

Rot ohne Transparenz: rgba(255,0,0, 1.0)

Blau mit Transparenz: rgba(0,0,255,0.3)

CSS – Farben

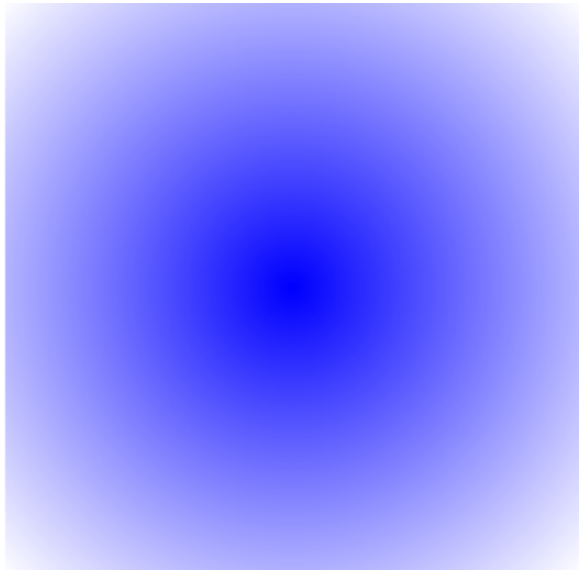
- HSLA(Farbton, Sättigung, Helligkeit, Alpha)
  - Farbton kann zwischen 0 und 359 liegen (in Anlehnung an ein 360° Farbrad)
  - Die Reihenfolge der Farben entspricht der eines Regenbogens
  - Rot, Orange, Gelb, Grün, Blau, Indigo, Violett
  - Der zweite Parameter gibt die Sättigung der Farbe in Prozent an (100% = voller Farbton und 0% = Grau)
  - 50% Helligkeit entspricht dem normalen Farbton. Je höher der Wert, umso heller die Farbe (100% = weiß und 0% = schwarz)
  - Hilfreiches Tool um RGB und HSL Werte zu ermitteln unter: http://www.workwithcolor.com/hsl-color-schemer-01.htm

Rot ohne Transparenz: hsla(0, 100%, 50%, 1)

Blau mit Transparenz: hsla(240, 100%, 50%, 0.3)

CSS – Farbverlauf

- Fließender Übergang von einer Farbe in die andere
  - Es gibt linearen und radialen Verlauf
  - Ein linearer Verlauf kann horizontal, vertikal oder diagonal sein
  - Kann auch über mehr als zwei Farben gehen



CSS – Farbverlauf

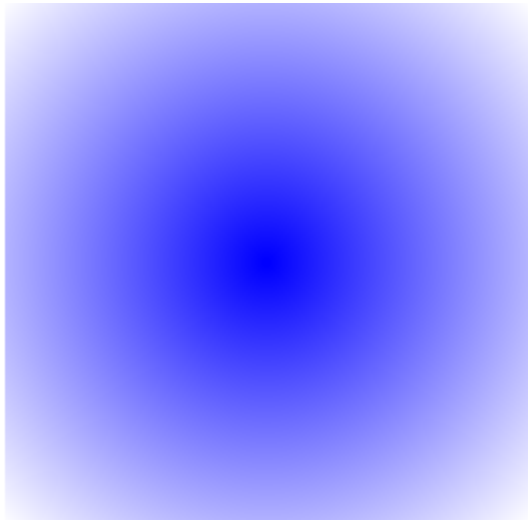
Linearer Verlauf

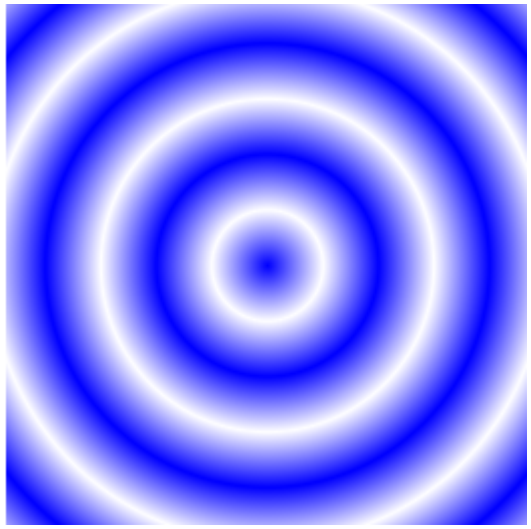
- Syntax
  - `background: linear-gradient(to bottom, white 0%, blue 100%);`
    - Der erste Parameter gibt die Verlaufsrichtung an (hier: von oben nach unten)
    - Die restlichen Parameter geben die Farben an
    - Optionale Prozentwerte geben die Stellen des Farbwechsels an
  - `background: linear-gradient(45deg , blue, white);`
    - Diagonale Verläufe können durch Winkelangaben realisiert werden

CSS – Farbverlauf

Radialer Verlauf

- Syntax
  - `background: radial-gradient(blue, white);`
    - Parameter für die Farbverteilung wie bei linearem Verlauf
  - `background: repeating-radial-gradient(blue, white 15%, blue 30%);`
    - Erzeugt wiederholten Farbverlauf
    - Hier: Verlauf von blau zu weiß, zurück zu blau für einen fließenden Übergang





CSS – Text-/Elementfluss

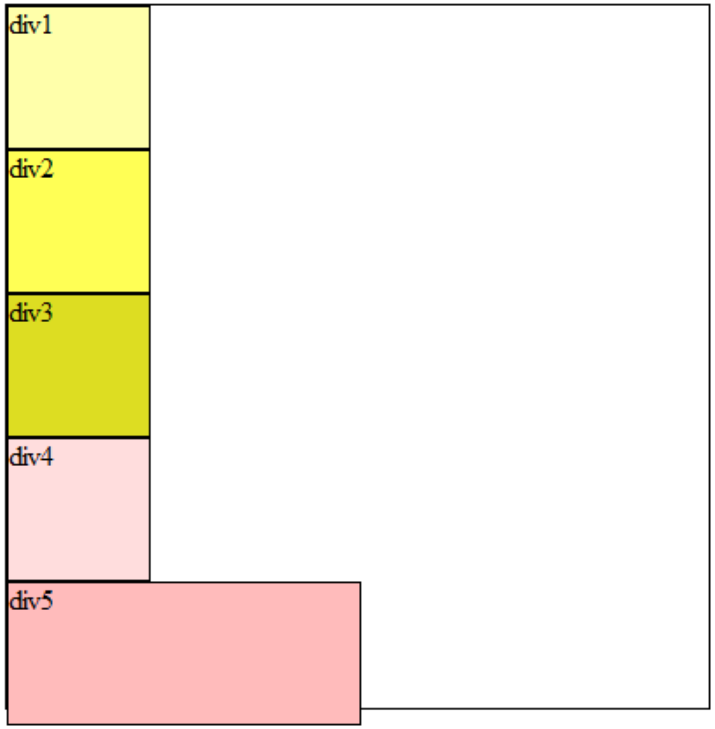
- Üblicherweise erscheinen Elemente untereinander

- Float-Attribut
  - Nimmt ein Element aus dem normalen Seitenfluss
  - `float:left` z.B. stellt das Element auf die linke Seite des Eltern-Elements
  - Alle anderen Elemente (Text oder andere Blocklevel-Elemente) fließen um dieses Element herum
  - Mögliche Werte `float:left, float:right, float:none`

- Clear-Attribut
  - Wird genutzt, um das *floating* zu beenden
  - Das erste Element, das nicht mehr um das floating-Element fließen soll, bekommt dieses Attribut
  - Mögliche Werte `clear:left, clear:right, clear:both`

CSS – Text-/Elementfluss

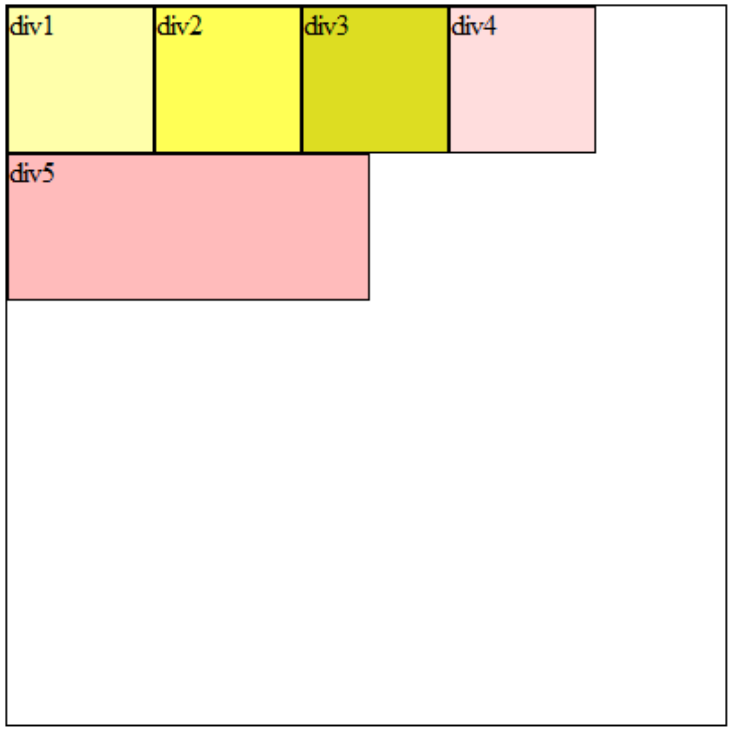
- Beispiel – Normaler Fluss




```
#div1 {
       width:80px; height:80px;
       background-color:#ffa; }
#div2 {
       width:80px; height:80px;
       background-color:#ff5; }
#div3 {
       width:80px; height:80px;
       background-color:#dd2; }
#div4 {
       width:80px; height:80px;
       background-color:#fdd; }
#div5 {
       width:200px; height:80px;
       background-color:#fbb; }
```

CSS – Text-/Elementfluss

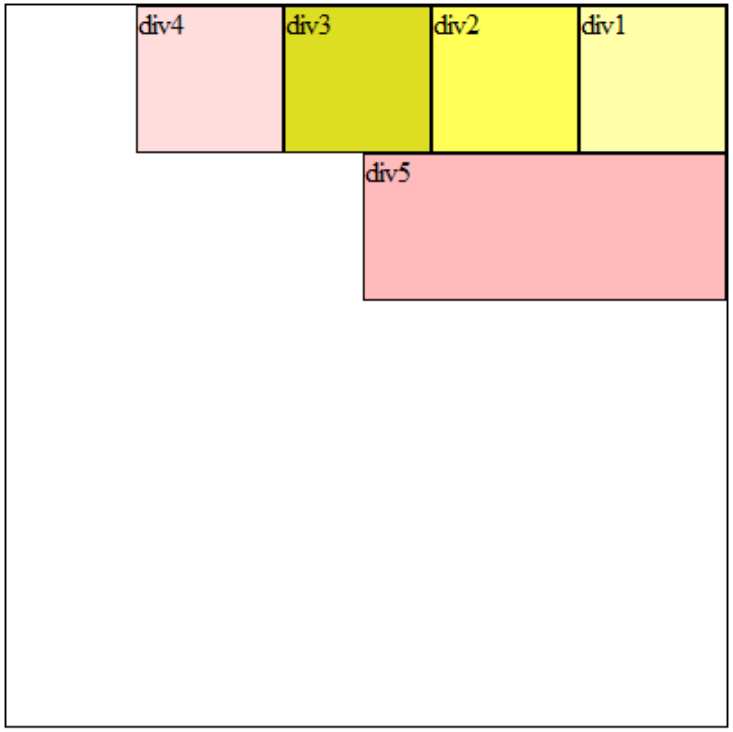
- Beispiel – Linksfluss




```
#div1 {float:left;
       width:80px; height:80px;
       background-color:#ffa; }
#div2 {float:left;
       width:80px; height:80px;
       background-color:#ff5; }
#div3 {float:left;
       width:80px; height:80px;
       background-color:#dd2; }
#div4 {float:left;
       width:80px; height:80px;
       background-color:#fdd; }
#div5 {float:left;
       width:200px; height:80px;
       background-color:#fbb; }
```

CSS – Text-/Elementfluss

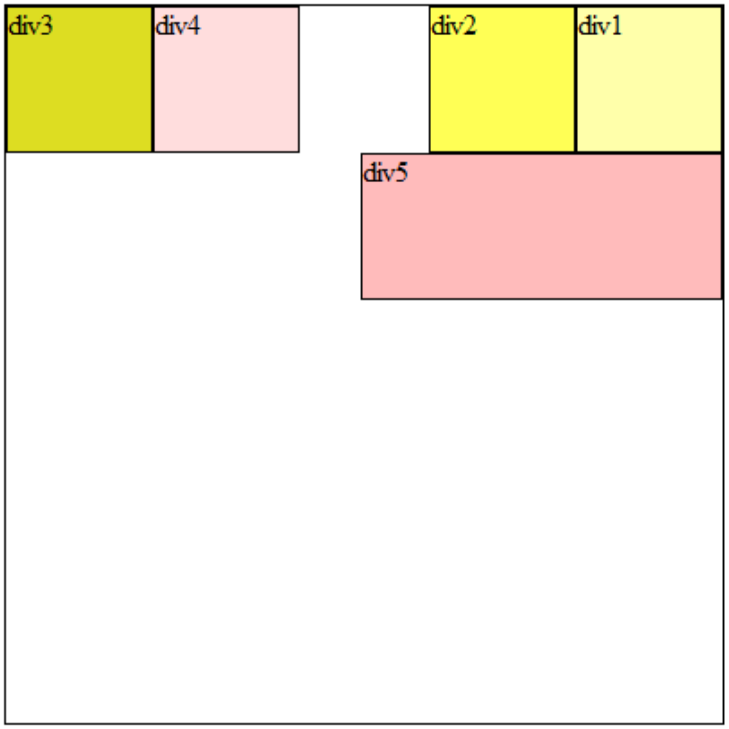
- Beispiel – Rechtsfluss




```
#div1 {float:right;
       width:80px; height:80px;
       background-color:#ffa; }
#div2 {float:right;
       width:80px; height:80px;
       background-color:#ff5; }
#div3 {float:right;
       width:80px; height:80px;
       background-color:#dd2; }
#div4 {float:right;
       width:80px; height:80px;
       background-color:#fdd; }
#div5 {float:right;
       width:200px; height:80px;
       background-color:#fbb; }
```

CSS – Text-/Elementfluss

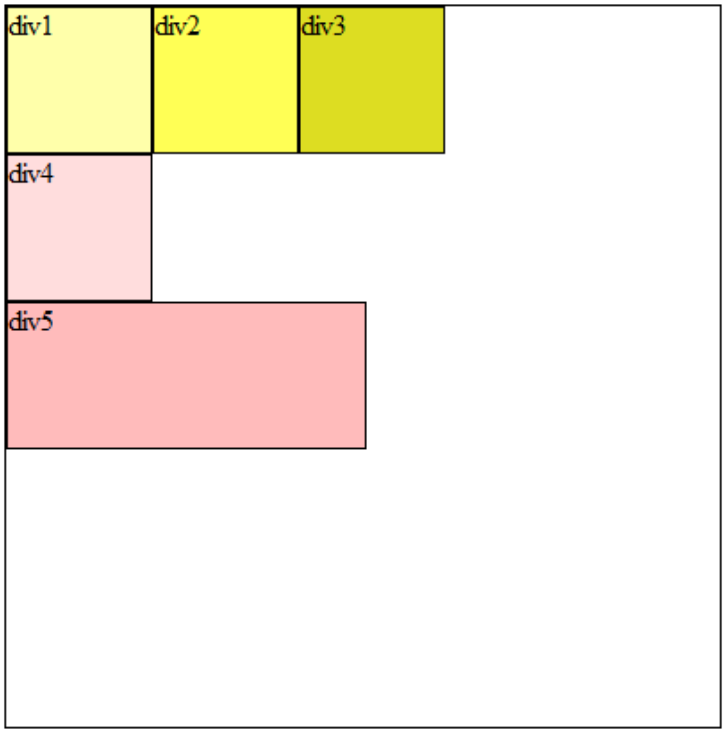
- Beispiel – Linksfluss und Rechtsfluss gemeinsam




```
#div1 {float:right;
       width:80px; height:80px;
       background-color:#ffa; }
#div2 {float:right;
       width:80px; height:80px;
       background-color:#ff5; }
#div3 {float:left;
       width:80px; height:80px;
       background-color:#dd2; }
#div4 {float:left;
       width:80px; height:80px;
       background-color:#fdd; }
#div5 {float:right;
       width:200px; height:80px;
       background-color:#fbb; }
```

CSS – Text-/Elementfluss

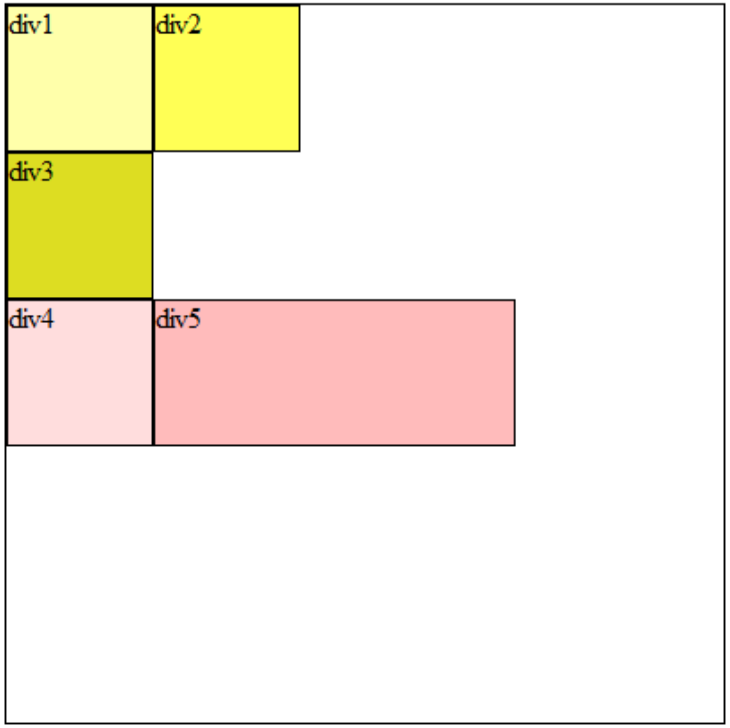
- Beispiel – Linksfluss mit Clear




```
#div1 {float:left;
       width:80px; height:80px;
       background-color:#ffa; }
#div2 {float:left;
       width:80px; height:80px;
       background-color:#ff5; }
#div3 {float:left;
       width:80px; height:80px;
       background-color:#dd2; }
#div4 {clear:left;
       width:80px; height:80px;
       background-color:#fdd; }
#div5 {
       width:200px; height:80px;
       background-color:#fbb; }
```

CSS – Text-/Elementfluss

- Beispiel – Linksfluss mit Clear-Unterbrechung




```
#div1 {float:left;
       width:80px; height:80px;
       background-color:#ffa; }
#div2 {float:left;
       width:80px; height:80px;
       background-color:#ff5; }
#div3 {clear:left;
       width:80px; height:80px;
       background-color:#dd2; }
#div4 {float:left;
       width:80px; height:80px;
       background-color:#fdd; }
#div5 {float:left;
       width:200px; height:80px;
       background-color:#fbb; }
```

CSS – Positionierung

Absolute Positionierung

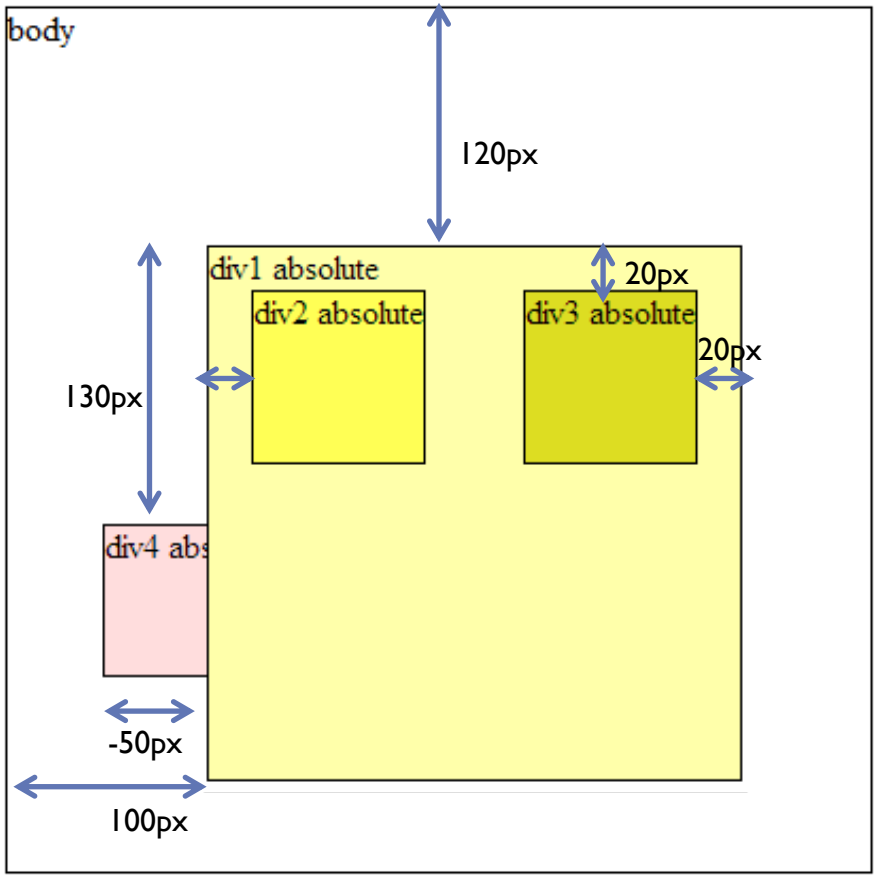
- Elemente erscheinen normalerweise in der Reihenfolge, in der sie im Code stehen
- Das Attribut `position` mit dem Wert `absolut` nimmt das Element aus dem normalen Fluss
- Das Element erscheint genau an der spezifizierten Stelle
- Die Angaben `top, left, right` oder `bottom` geben die entsprechenden Abstände zum Rand an
- Bei `position:absolute` beziehen sich die Abstände zum Rand des Eltern-Blocklevel-Elements
- Z-Index ermöglicht es Elemente zu Überlappen. Default-Wert ist 0
- Elemente mit höherem z-Index erscheinen vor Elementen mit niedrigem z-Index

```
<body> body
   <div id="div1">div1 absolute
     <div id="div2">div2 absolute</div>
     <div id="div3">div3 absolute</div>
     <div id="div4">div4 absolute</div>
   </div>
</body>
```

siehe Beipiel

CSS – Positionierung

- Absolute Positionierung - Beispiel




```
body  { width:400px; height:400px;}
#div1 { position:absolute;
        top:120px; left:100px;
        width:250px; height:250px;
        z-index:3;
        background-color:#ffa; }
#div2 { position:absolute;
        top:20px; left:20px;
        width:80px; height:80px;
        z-index:1;
        background-color:#ff5; }
#div3 { position:absolute;
        top:20px; right:20px;
        width:80px; height:80px;
        z-index:2;
        background-color:#dd2; }
#div4 { position:absolute;
        top:130px; left:-50px;
        width:190px; height:70px;
        z-index:-1;
        background-color:#fdd; }
```

CSS – Positionierung

- Relative Positionierung
  - Das Element mit dem Attribut `position:relative` bleibt im normalen Fluss der Seite
  - Die Position wird relativ zur Position anderer Elemente um ihn bestimmt
  - Alternativ ausgedrückt: Ausgangspunkt für die neue Position ist die ursprüngliche Position des Elements im Seitenfluss

```
<body>
   <p>The text at the end of this sentence
      <span class="super">is in superscript</span></p>
   <p>The text at the end of this sentence
      <span class="sub">is in subscript</span></p>
   <p>The text at the end of this sentence
      <span class="shiftleft">is shifted left</span></p>
   <p>The text at the end of this sentence
      <span class="shiftright">is shifted right</span></p>
</body>
```

CSS – Positionierung

- Relative Positionierung - Beispiel

```
.super       {position: relative; top: -1ex; }
.sub       {position: relative; bottom: -1ex; }
.shiftleft   {position: relative; left: -1ex; }
.shiftright  {position: relative; right: -1ex; }
```

The text at the end of this sentence is in superscript

The text at the end of this sentence is in subscript

The text at the end of this sentenceis shifted left

The text at the end of this sentence is shifted right


CSS
Global
HD

CSS – Animation

- CSS-Animationen ohne Flash oder JavaScript
  - Wurden erst durch CSS3 ermöglicht
  - Sind für fast alle HTML-Elemente möglich
  - Zusätzliche Software wird nicht benötigt

- Animationen überführen ein Element fließend von einem Zustand in den nächsten
  - Dafür müssen Keyframes festgelegt werden
  - Keyframes beschreiben einzelne Zustände in einer Animation
  - Eine Animation enthält beliebig viele festgelegte Zustände
  - Es können folglich beliebig viele CSS-Regeln beliebig oft geändert werden

CSS – Animation

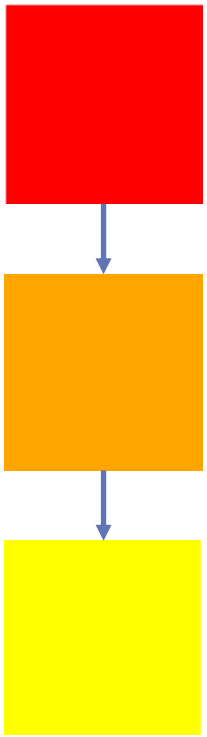
- Die `@keyframes`-Regel
  - Legt genau fest, zu welchen Zeitpunkten in der Animation, welche CSS-Regeln gelten
  - Die fließenden Transitionen zwischen den festgelegten Zuständen geschehen automatisch
  - Die so festgelegte Animation muss anschließend in ein Element eingebunden werden

```
@keyframes beispiel {
    from {background-color: red;}
    to {background-color: yellow;}
}
```

Definition der Animation

```
div {
    width: 100px;
    height: 100px;
    background-color: yellow;
    animation-name: beispiel;
    animation-duration: 4s;
}
```

Einbindung der Animation

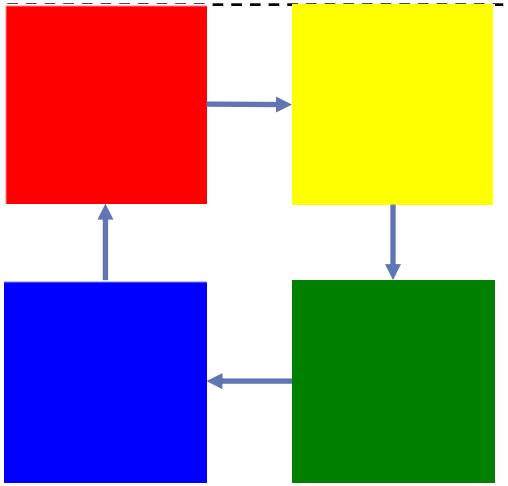


CSS – Animation

- Keyframes Syntax
  - Für mehr als nur zwei Zustände werden statt `from` und `to` Prozentwerte angegeben
  - Innerhalb der geschweiften Klammern können beliebig viele CSS Regeln festgelegt werden

```
@keyframes beispiel {
    0%   {background-color: red; left: 0px; top: 0px;}
    25%  {background-color: yellow; left: 200px; top: 0px;}
    50%  {background-color: green; left: 200px; top: 200px;}
    75%  {background-color: blue; left: 0px; top: 200px;}
    100% {background-color: red; left: 0px; top: 0px;}
}
```

Hintergrundfarbe und Position des Quadrates gehen fließend ineinander über



CSS – Animation

- Einbindung Syntax
  - Das Element, in dem die Animation eingebunden wurde, kann durch verschiedene Attribute die Animation weiter modifizieren
  - `animation-name: beispiel` legt fest, welche Animation benutzt wird
  - `animation-duration: 4s` legt fest, wie lange ein Durchlauf der Animation dauert. Dieser Wert sollte gesetzt werden, da der Standardwert 0 ist (keine Animation)
  - `animation-delay: 2s` legt fest, wie lange gewartet werden soll, bis die Animation beginnt
  - `animation-iteration-count: 3` legt fest, wie oft die Animation ausgeführt werden soll. Falls sie niemals aufhören soll, kann hier der Wert `infinite` angegeben werden
  - `animation-direction: reverse` legt fest, dass die Animation in umgekehrter Reihenfolge durchgeführt werden soll. Der Wert `alternate` gibt hierbei an, dass die Animation vorwärts, rückwärts, vorwärts u.s.w. ausgeführt werden soll
  - `animation-timing-function:` legt die Geschwindigkeitskurve der Animation fest
    - `ease` Animation startet langsam, wird schneller und dann wieder langsamer (default)
    - `linear` durchgehend die selbe Animationsgeschwindigkeit
    - `ease-in` langsamer Animationsstart
    - `ease-out` langsames Animationsende
    - `ease-in-out` langsamer Animationsstart und langsames Animationsende

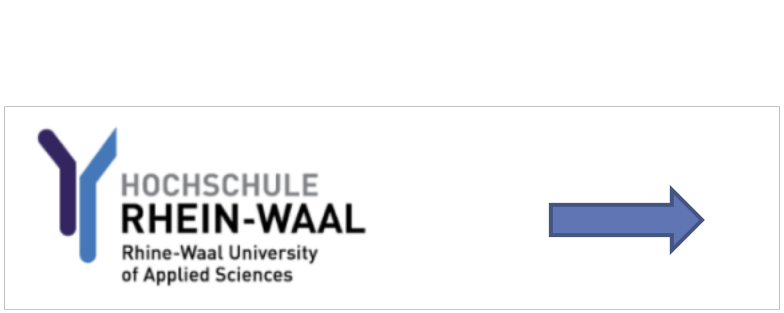
CSS – Animation

- Beispiel

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    width: 300px;
    height: 100px;
    background-image: url(logo_tu.png);
    background-repeat: no-repeat;
    position: relative;
    animation-name: slide;
    animation-duration: 3s;
}
@keyframes slide {
    from {left: 200px; opacity: 0.0;}
    to {left: 0px; opacity: 1.0;}
}
</style>
</head>
<body>
    <div></div>
</body>
</html>
```

CSS – Transition und Transformation

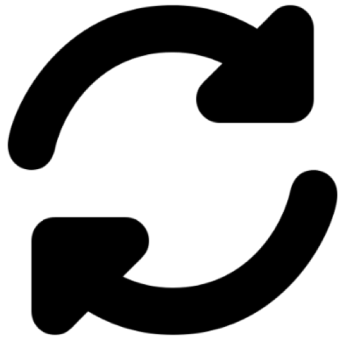
- Transformationen
  - Erlauben das Bewegen, Rotieren und Skalieren von Elementen (nicht animiert)
  - `transform: rotate(180deg) scale(2, 3) skewX(20deg);`
  - Das Element dreht sich um 180 Grad (rotate(180deg))
  - Verdoppelt seine Breite und verdreifacht die Höhe (scale(x, y))
  - Wird horizontal (x-Achse) um 20 Grad verzerrt
    (Analog: SkewY(20deg), vertikal verzerrt (y-Achse))



CSS – Transition und Transformation

- Transitionen
  - Erlauben es, CSS Eigenschaftswerte fließend ineinander übergehen zu lassen
  - Beispiel: Beim Mouseover rotiert das Bild um 360°

```
div {
    width: 50px;
    height: 50px;
    background-image: url(pfeil.png);
    transition: transform 3s;
}
div:hover {
    transform: rotate(360deg);
}
```



  - Verlässt die Maus das Element, so geht es fließend wieder in den alten Zustand über
  - So lassen sich z.B. animierte Buttons oder Firmenlogos erstellen
  - Im Gegensatz zu Animationen lassen sich bei Transitionen keine Zwischenpunkte (keyframes) festlegen und keine Wiederholungen angeben. Transitionen müssen über ein Ereignis wie mouseover (:hover) ausgelöst werden.

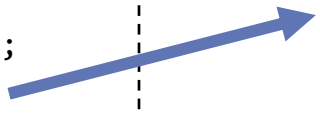
CSS – Media Types

- Unterschiedliche Darstellungen der selben Seite
  - Verschiedene Endgeräte (Monitor, Smartphone, Drucker…) erfordern ggf. unterschiedliche Darstellungen
    - In der Regel sind auf Bildschirmen größere Buchstaben und serifenfreie Schriftarten sinnvoll, wohingegen die Druckansicht eher kleinere Buchstaben und Serifen benötigt
  - MediaTypes ermöglichen es, unterschiedliche Style-Regeln für verschiedene Endgeräte im selben Stylesheet zu definieren
- Die `@media`-Regel
  - Definiert die unterschiedlichen CSS-Regeln für das jeweilige Endgerät
  - `@media all` Gilt für alle Typen von Endgeräten
  - `@media screen` Gilt für Computerbildschirme
  - `@media print` Gilt für Drucker und Druckansichten
  - `@media handheld` Gilt für Handheld-Geräte

CSS – Media Types

- Beispiel

```
<!DOCTYPE html> <html> <head>
<style>
@media screen {
    p {
        font-family: verdana, sans-serif;
        font-size: 17px;
        color: blue;
    }
}
@media print {
    p {
        font-family: georgia, serif;
        font-size: 14px;
}
}
</style> </head>
<body>
    <h2>Die @media-Regel</h2>
    <p>Dieser Textabsatz wird in Druck-
    und Normalansicht in unterschiedlicher
    Schrift, Schriftgröße und Farbe
    dargestellt</p>
</body> </html>
```

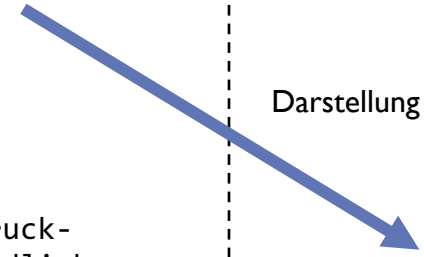


Die @media-Regel

Dieser Textabsatz wird in Druck- und Normalansicht in unterschiedlicher Schrift, Schriftgröße und Farbe dargestellt

Darstellung Computerbildschirm

Darstellung Drucker



Die @media-Regel

Dieser Textabsatz wird in Druck- und Normalansicht in unterschiedlicher Schrift, Schriftgröße und Farbe dargestellt

CSS Variable

- Mit custom properties können Variable definiert und später für Deklarationen verwendet werden.
- Bsp.: Farbe entsprechend Unternehmens-CI festlegen ...

```
:root {
  --hauptfarbe: #0000ff;
  --zweitfarbe: #0022cc;
}
```

`root` ist eine Pseudoklasse, die den Gültigkeitsbereich der Deklaration auf das gesamte HTML-Dokument festlegt.

- ... und mit der Funktion `var()` verwenden

```
#logo {
  color: var(--hauptfarbe);
  background-color: var(--zweitfarbe);
}

.text {
  color: var(--hauptfarbe);
}
```

CSS – Präprozessoren

- CSS hat viele Nachteile
  - Redundanter Code für unterschiedliche Selektoren
  - Keine Imports
  - Keine Verschachtelung von Selektoren in den Regel-Blöcken, z.B.:

```
.Klasse1{
    margin: 10px;
    .Klasse2{
        padding:2px;
    }
}
```

- CSS-Präprozessoren
  - Viele Alternativen: SASS, LESS, Stylus, Turbine etc.
  - In dieser Vorlesung: SASS (Syntactically Awesome Stylesheets) mit der SCSS-Syntax (Sassy CSS)

CSS – SASS

Einleitung

- SASS Anweisungen in einer ".scss"-Datei
  - Aufteilung in mehrere Dateien möglich (Import)
- Volle CSS-Kompabilität
  - Reines CSS in scss-Dateien ebenfalls erlaubt
    - Nutzung nur solcher SASS-Komponenten, die benötigt werden
- In Anwendung
  - Schreiben des SASS-Codes
  - Nutzung eines Präprozessors-Tools z.B. "Prepros" http://prepros.io oder des Online-Präprozessors http://sassmeister.com
  - Einbinden der generierten CSS-Datei in das HTML-Dokument

CSS – SASS

Beispiele

- Verschachtelung

```scss
/* standard.scss */
nav {
    margin: 0;
    ul {
        padding: 0;
        list-style: none;
    }
    li {
        display: inline-block;
    }
    a {
        display: block;
        padding: 6px 12px;
        text-decoration: none;
    }
}
```

```css
/* standard.css */
nav {
    margin: 0;
}
nav ul {
    padding: 0;
    list-style: none;
}
nav li {
    display: inline-block;
}
nav a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
}
```

- **$name** (Variable definieren)

```scss
/* standard.scss */
$font-stack: Helvetica, sans-serif;
$primary-color: #333;
body {
    font: 100% $font-stack;
    color: $primary-color;
}
```

```css
/* standard.css */
body {
    font: 100% Helvetica, sans-serif;
    color: #333;
}
```


Beispiele von http://sass-lang.com/guide und "CSS" von Kai Laborenz

CSS – SASS

Beispiele

- **@import** (Importieren)
  - Dateiname der ausgelagerten Datei beginnt mit einem Unterstrich

```scss
/* standard.scss */
@import 'reset';
body {
    font: 100% Helvetica, sans-serif;
    background-color: #efefef;
}
```

```scss
/* _reset.scss */
html, body, ul, ol {
    margin: 0;
    padding: 0;
}
```

```css
/* standard.css */
html, body, ul, ol {
    margin: 0;
    padding: 0;
}
body {
    font: 100% Helvetica, sans-serif;
    background-color: #efefef;
}
```

- **@mixin** (Wiederverwendung von Code-Blöcken)
  - Können als Methoden mit Parametern angesehen werden

```scss
/* standard.scss */
@mixin border-radius($radius) {
    border-radius: $radius;
}

.box {
    @include border-radius(10px);
}
```

```css
/* standard.css */
.box {
    border-radius: 10px;
}
```


Beispiele von http://sass-lang.com/guide und "CSS" von Kai Laborenz

CSS – SASS

Beispiele

- Operatoren (+, -, *, /, %)
  - Achtung, inkompatible Typen beachten: (20px + 10px) * 20%

```scss
/* standard.scss */
.galleryWrapper {
    width: 100%;
}
.galleryWrapper .galleryItem {
    float: left;
    width: 320px / 960px * 100%;
}
```

```css
/* standard.css */
.galleryWrapper {
    width: 100%;
}
.galleryWrapper .galleryItem {
    float: left;
    width: 33.33333%;
}
```

```html
<!-- Beispiel.html -->
<html><head><style>
    .galleryWrapper {
        width: 100%;
    }
    .galleryWrapper .galleryItem {
        float: left;
        width: 33.33333%;
     }
</style></head><body>
    <div class="galleryWrapper">
        <img src="http://lorempixel.com/200/200/technics/1" class="galleryItem">
        <img src="http://lorempixel.com/200/200/technics/2" class="galleryItem">
        <img src="http://lorempixel.com/200/200/technics/3" class="galleryItem">
        <img src="http://lorempixel.com/200/200/technics/4" class="galleryItem">
        <img src="http://lorempixel.com/200/200/technics/5" class="galleryItem">
    </div>
</body></html>
```

Beispiele von http://sass-lang.com/guide und "CSS" von Kai Laborenz

CSS – SASS

Beispiele

- **@extend** (Vererbung)

```
/* standard.scss */
.message {
    border: 1px solid #ccc;
    padding: 10px;
    color: #333;
}
.success {
    @extend .message;
    border-color: green;
}
.error {
    @extend .message;
    border-color: red;
}
```

```
/* standard.css */
.message, .success, .error {
    border: 1px solid #ccc;
    padding: 10px;
    color: #333;
}
.success {
    border-color: green;
}
.error {
    border-color: red;
}
```

- Schleifen und Kontrollstrukturen
  - Siehe z.B.: http://florenz.co.uk/schleifen-und-bedingte-anweisungen-in-sass-for-each-while-if

Beispiele von http://sass-lang.com/guide und "CSS" von Kai Laborenz

CSS – Frameworks

Einleitung

- Bestimmte CSS-Eigenschaften und Strukturelemente sind für den Stil aller Websites sinnvoll
  - Hervorhebungen für Quellcode, Hinweise etc.
  - Icons zum Weiterblättern (Pagination)
  - Tooltips, Dialoge, etc.
  - Weitere Beispiele: http://getbootstrap.com/examples/theme

- Erstellung eigener CSS-Bibliothek
  - Wiederverwendbare CSS-Regeln → Implementierung des eigenen Stils
  - Zeitintensiv bei Erstellung und Pflege
  - Browser- und Abwärtskompatibilitäten zu berücksichtigen

- Daher: Nutzung von fertigen Frameworks, die die Grundgestaltung übernehmen
  - Z.B.: Foundation, Bootstrap, Skeleton, YAML, etc.

CSS – Frameworks

Bootstrap – Einführung

- Bootstrap wurde 2011 von Twitter veröffentlicht
  - http://getbootstrap.com
- Nutzt HTML, CSS und JavaScript
  - Benötigt JQuery-Bibliothek (s. Kapitel JavaScript)

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script>
```

- Vordefinierte Klassen und Komponenten
- "Mobile first"-Ansatz
  - Automatische Anpassung des Layouts auf kleinen Displays
- Schnelle Entwicklung von "gutaussehenden" Websites

- Einbindung des Frameworks (extern)

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap-theme.min.css">
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script>
```

CSS – Frameworks

Bootstrap – Buttons

- Hinzufügen der Klasse "`btn btn-default`" lässt Element wie einen Button aussehen
- Verschiedene Farbgestaltungen möglich

```
<html><head>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap-theme.min.css">
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script>
</head><body>
    <a class="btn btn-default" href="http://www.hsrw.eu">Link</a>
    <span class="btn btn-default">Default</span>
    <span class="btn btn-primary">Primary</span>
    <span class="btn btn-success">Success</span>
    <span class="btn btn-info">Info</span>
    <span class="btn btn-warning">Warning</span>
    <span class="btn btn-danger">Danger</span>
    <span class="btn btn-link">Link</span>
</body></html>
```

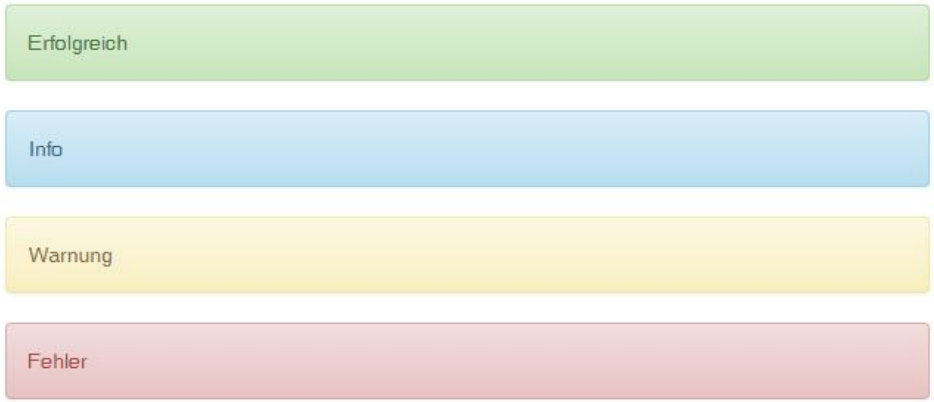
http://getbootstrap.com/css/#buttons

CSS – Frameworks

Bootstrap – Hinweise

- Hinzufügen der Klasse "`alert alert-success`" zu einem **div**-Tag lässt das Tag wie einen Hinweis aussehen
- Verschiedene Farbgestaltungen möglich

```
<div class="alert alert-success">Erfolgreich</div>
<div class="alert alert-info">Info</div>
<div class="alert alert-warning">Warnung</div>
<div class="alert alert-danger">Fehler</div>
```



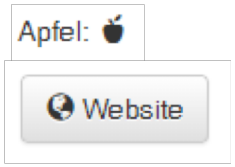
http://getbootstrap.com/components/#alerts

CSS – Frameworks

Bootstrap – Icons

- Verwendung von Icons sinnvoll, um Semantik zu verdeutlichen
- Nutzung mittels **span**-Tag und der Klasse "`glyphicon glyphicon-T`" (T ist der Icon-Typ)
- Bootstrap bietet über 250 Icons in Form einer Schriftart
  - Verlustfreie Skalierung
  - Nutzung mit Buttons und anderen Elementen möglich
  - Übersicht über verfügbare Icons: http://getbootstrap.com/components/#glyphicons

```
Apfel: <span class="glyphicon glyphicon-apple"></span>
<br>
<a class="btn btn-default" href="http://hsrw.eu">
    <span class="glyphicon glyphicon-globe"></span> Website
</a>
```



http://getbootstrap.com/components/#glyphicons

CSS – Framworks

Bootstrap – Komponenten

- Grid-System zur einfachen Erstellung von Layouts
  - Anpassung je nach Displaygröße: "Responsive Design"
  - Standardmäßig 12 Spalten verfügbar
  - Bei zu wenig Platz: Spalten werden in Zeilen umgebrochen
- Syntax
  - Umgebendes **div**-Element mit der Klasse: `row`
  - Innere **div**-Elemente (Spalten) mit Klasse: `col-K-B`
    - K: Geräteklasse
    - B: Breite der Spalte (1-12)
    - Bsp.: "`col-md-6`"
    - Mehrere Geräteangaben je **div**-Tag möglich

Gerät	Größe	Präfix
Sehr klein (Smartphone)	< 768px	`col-xs-`
Klein (Tablet)	>= 768px	`col-sm-`
Medium (Desktop)	>= 992px	`col-md-`
Groß (Desktop)	>= 1200px	`col-lg-`

http://getbootstrap.com/css/#grid

Bootstrap Container

- Container bilden die Basis des Grid-Systems in bootstrap
- Zeilen (rows) und darin enthaltene Spalten (cols) werden in „Container“ gefasst
- Container sind responsiv und wahlweise mit fixierter Breite (fixed-width) oder voller Breite (fluid-width)
- .container => Rand links und rechts
- .container-fluid => 100% breit, auch bei verschiedenen Geräten

CSS – Frameworks

Bootstrap – Komponenten

- Beispiel zum Grid-System
  - Geräteklasse: Groß

  - Geräteklasse: Medium

  - Geräteklasse: Kleiner als medium

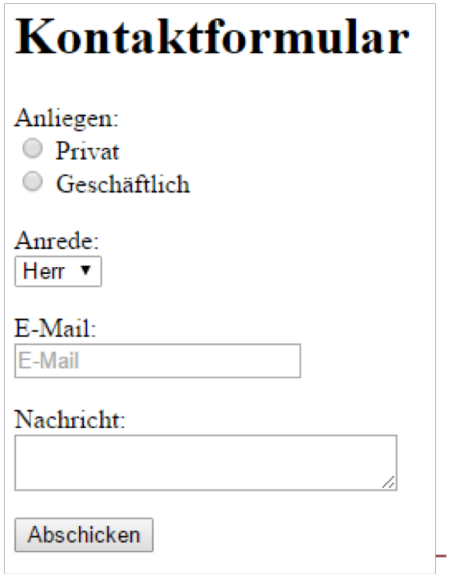

```
<div class="container">
    <div class="row">
        <div class="col-lg-2">Navigations-Spalte</div>
        <div class="col-lg-10">
            <div class="row">
                <div class="col-lg-8 col-md-6">
                    <h1>Überschrift</h1>
                </div>
                <div class="col-lg-4 col-md-6">
                    <h2>Weitere Informationen</h2>
                </div>
            </div>
        </div>
    </div>
</div>
```


http://getbootstrap.com/css/#grid

CSS – Frameworks

Bootstrap – Weitere Komponenten

- Viele weitere Komponenten verfügbar
  - Labels, Button-Gruppen, etc.
  - Spezielle Effekte benötigen JavaScript (hier nicht behandelt)
    - Dialoge, Tooltips, Bilder-Gallerien, Tabs, etc.
- Auch durch ledigliche Einbindung bereits Vorteile
  - Schriftarten, Abstände, grundlegendes Aussehen, uvm. bereits für die Standard-Tags definiert
    - Bild links: ohne Bootstrap
    - Bild Rechts: mit Bootstrap



Weiterführende Dokumentation zu Bootstrap

- https://getbootstrap.com/docs/4.0/layout/overview/

CSS – Frameworks

Bootstrap – Vor- & Nachteile

- Vorteile
  - Zeiteinsparung durch vorgegebenen Stil
  - Viele Elemente wie Buttons, Tooltips usw. bereits vorhanden
  - Einheitliches browserübergreifendes Design
  - Erweiterbarkeit und Anpassbarkeit durch *sass*-Quellcode

- Nachteile
  - Einarbeitungszeit
  - Größe des Frameworks → Höhere Ladezeit
  - Nutzung zu vieler Features, kann zu "verspielten" Websites führen
  - Das HTML-Layout ist stark an das Framework gebunden (Austausch schwierig)