Fortgeschrittenes CSS und SASS

Fortgeschrittenes CSS & SASS

Jannis Seemann

Fortgeschrittenes CSS und SASS

- **Wer bin ich überhaupt?**
  - Studierter Informatiker (TU München)
  - Webentwickler aus Leidenschaft
  - Und Dozent für Online-Kurse
- **Warum diesen Kurs?**
  - Der Einstieg in CSS ist recht einfach
  - CSS ist aber unglaublich mächtig
  - Es macht daher Sinn, dass wir uns damit mal genauer beschäftigen
  - **Mein Ziel:** Nach Abschluss dieses Kurses beherrscht du CSS
  - Zudem kannst du CSS-Tricks verwenden, die dir später sehr viel Zeit sparen werden

Wie ist dieser Kurs aufgebaut?

- **Projekt 1: Einfache Landing-Page**
  - Animationen
  - Farbverläufe (Gradients)
- **Projekt 2: Komplexes Web-Projekt**
  - **CSS-Expertenwissen:** Wie funktioniert CSS "unter der Haube"? Was hat das für Auswirkungen für denen CSS-Code?
  - **SASS / SCSS:** Behalte auch bei vielen Zeilen CSS-Code den Überblick
  - **Moderne CSS-Features:**
    - Flexbox
    - CSS-Grid
    - 3D-Effekte in CSS
    - Interaktive Webseiten nur mit CSS
  - Bonus: Ladezeitoptimierung (SVG-Sprite, das Picture-Element)

Fortgeschrittenes CSS und SASS

CSS: Fortgeschrittene Grundlagen

Jannis Seemann

Fortgeschrittenes CSS und SASS

CSS: Fortgeschrittene Grundlagen

Jannis Seemann

Fortgeschrittenes CSS und SASS

Was erwartet dich? Animationen in CSS

Jannis Seemann

Animationen in CSS

- Was erwartet dich?
  - Wir legen zuerst unser erstes Kurs-Projekt an
  - Und erstellen dort grundlegenden HTML- und CSS-Code
- Anschließend schauen wir uns Animationen an:
  - Transitions (z.B. bei Hover)
  - Kompliziertere Animationen mit @keyframes

Fortgeschrittenes CSS und SASS

Animationen in CSS

Jannis Seemann

Animationen in CSS

- In CSS gibt es verschiedene Möglichkeiten, Animationen auszuführen
- **Einfachste Möglichkeit:**
  - Wir ändern eine Eigenschaft nach Zustand ab, z.B. bei :hover
  - Standardmäßig werden diese Eigenschaften dann direkt geändert
  - Aber wir können diese Änderung auch animieren lassen:

```css
.button {
  background-color: white;
  transition: background-color 2s;
}
.button:hover {
  background-color: green;
}
```

Fortgeschrittenes CSS und SASS

Animationen in CSS

Jannis Seemann

Transition-Timing-Function

- Die Transition-Timing-Function definiert, wie die Animation ablaufen soll
- Hier nutzen wir i.d.R. „ease-in-out“ oder „linear“
- Wir haben aber auch andere Möglichkeiten
- Insbesondere können wir über cubic-bezier sehr genau steuern, wie die Animation ablaufen soll - mehr dazu gleich im Beispiel
- **Beispiel:**

```
.button {
  background-color: white;
  transition: background-color 2s ease-in-out 1s;
}
.button:hover {
  background-color: green;
}
```

Beispiel: Dokumentation von CSS

- Das ist eine gute Gelegenheit, uns mal die Dokumentation anzuschauen:
  - caniuse:
    - => https://caniuse.com/?search=transition
  - Offizielle CSS-Spezifikation
    - => https://www.w3.org/TR/css-transitions-1

Fortgeschrittenes CSS und SASS

Animationen in CSS (Keyframes)

Jannis Seemann

Keyframe-Animationen in CSS

- Eine weitere Möglichkeit sind sog. Keyframe-Animationen
- Diese erlauben uns, die Animation sehr viel genauer zu steuern
- Die Schreibweise ist aber aufwendiger
- Zuerst müssen Keyframes definiert werden
- **Beispiel:**

```
@keyframes animation-color {
  0%, 100% {
    color: green;
  }
  33% {
    color: red;
  }
  66% {
    color: yellow;
  }
}
```

Keyframe-Animationen in CSS

- Darauf aufbauend können wir jetzt diese Animation verwenden:

```css
h1 {
  animation-name: animation-color;
  animation-duration: 1s;
  animation-timing-function: ease-in-out;
  animation-iteration-count: 1;
}

@keyframes animation-color {
  0%, 100% {
    color: green;
  }
  50% {
    color: red;
  }
}
```

Fortgeschrittenes CSS und SASS

Best-Practices für Animationen

Jannis Seemann

Best-Practices: Animationen

- **Performance-Tipps:**
  - Animationen können sehr rechenaufwendig sein
  - Gerade auf älteren Smartphones / langsamen Desktop-Computern kann dies zu Problemen führen
  - 30-60 Bilder pro Sekunde
  - Pro Bild („Frame“) ca. 16,6 Millisekunden Zeit
  - Wie können wir sicherstellen, dass die Animation flüssig abläuft?

Best-Practices: Animationen

- **Einige CSS-Eigenschaften sind für den Browser viel Rechenarbeit:**
  - **Beispiel font-size:**
    - Hier muss der gesamte Text neu gelayouted werden
  - **Beispiel margin-top:**
    - Bei einem margin-top verschieben sich auch die Elemente danach u.U. weiter nach unten. Der Browser muss also das alles ausrechnen
  - **Beispiel color:**
    - Hier muss die Schriftfarbe geändert werden. Dabei müssen u.U. Sub-Pixel angepasst werden - aufwendig für den Browser!
- **Andere Eigenschaften können hingegen sehr schnell animiert werden:**
  - **Transform:**
    - Diese Befehle sind so optimiert, dass sie möglichst gut auf der Grafikkarte ausgeführt werden können
  - **Opacity:**
    - Die Sichtbarkeit (Gegenteil von Transparenz) kann auch gut auf die Grafikkarte ausgelagert werden
- Weitere Informationen
  - => https://csstriggers.com/

Fortgeschrittenes CSS und SASS

Was erwartet dich: Hintergründe in CSS

Jannis Seemann

Was erwartet dich: Hintergründe in CSS

- Die Reihenfolge von Hintergründen (background-color, background-image, background)
- Wie funktionieren Farbverläufe?
- Wie legen wir einen Farbverlauf über ein Bild?
- Richtig cooler Bonus: Schicker Button-Effekt für unser Projekt

Fortgeschrittenes CSS und SASS

Hintergründe in CSS

Jannis Seemann

Hintergründe in CSS

- **Ein HTML-Element kann verschiedene backgrounds haben:**
  - Zuerst eine Farbe (background-color)
  - Und dann eine oder mehrere Hintergrund-Bilder (background-image)
  - Diese Reihenfolge ist fest, die Hintergründe werden immer in dieser Reihenfolge platziert
- Statt den Spezialbefehlen können wir auch einfach die "background"-Eigenschaft in CSS verwenden
- **Aber Achtung:**
  - Auch hier muss die Reihenfolge beachtet werden!
- **Warum ist dies für dich wichtig?**
  - Hintergrundbild "abdunkeln"
  - Mehrere Farbverläufe kombinieren

Fortgeschrittenes CSS und SASS

Besondere Hintergründe mit CSS

Jannis Seemann

Besondere Hintergründe mit CSS

- Wir können auch Farbverläufe vom Browser generieren lassen
- **Wichtig:**
  - Diese Farbverläufe zählen nicht als Hintergrundfarbe, sondern als Hintergrundbild
  - => Dies wird später noch wichtig werden!
- **Farbübergänge können für uns generiert werden:**
  - linear-gradient([richtung], [farbe] [position],...)
  - radial-gradient([form] [position], [farbe] [position],...)
- Das sollten wir uns aber in der Praxis mal genauer anschauen!

Fortgeschrittenes CSS und SASS

CSS: Fortgeschrittene Grundlagen

Jannis Seemann

Fortgeschrittenes CSS und SASS

Was erwartet dich: CSS-Grundlagen

Jannis Seemann

CSS-Grundlagen

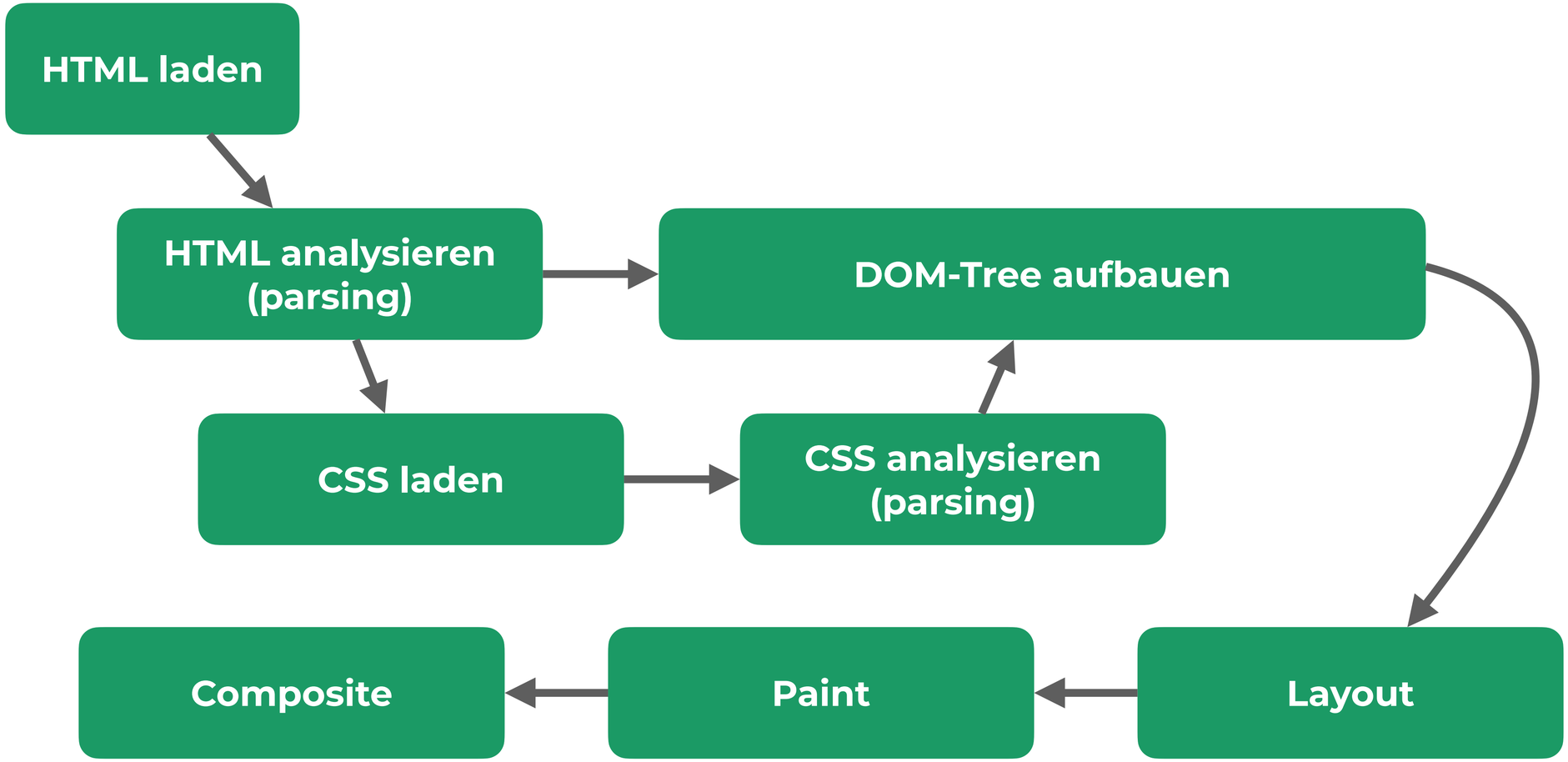
- **Was erwartet dich?**
  - Wie wird CSS überhaupt eingelesen und vom Browser verarbeitet?
  - Welche Regel gilt, wenn mehrere Regeln sich gegenseitig ausschließen?
  - Wie wird die Breite bzw. Höhe von einem Element definiert? Beinhaltet die width das padding oder nicht? Wie können wir das einstellen?
  - Wie berechnet sich die Schriftgröße?
  - Was ist der Unterschied zwischen Inline- und Block-Elementen
  - Responsive Design mit @media-Queries

Fortgeschrittenes CSS und SASS

Wie funktioniert CSS?

Jannis Seemann

Wie wird überhaupt eine Webseite angezeigt?



Wie wird eine Webseite angezeigt (vereinfacht)?

1. Aus dem HTML wird der DOM-Tree aufgebaut
2. Aus dem CSS wird der CSSOM-Tree aufgebaut
3. DOM-Tree und CSSOM-Tree werden zu einem Render-Tree kombiniert
4. Das Layout des Render-Trees wird berechnet
5. Dann wird der Render-Tree gezeichnet ("paint")
6. Und anschließend wird das "Compositing" ausgeführt

Quelle: https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-tree-construction

Fortgeschrittenes CSS und SASS

Wie wird CSS verarbeitet?

Was ist eine CSS-Regel?

CSS-Selector:

```
.main-content h1 {
    color: white;
    margin-bottom: 10px;
}
```

CSS-Deklaration

Aber was passiert, wenn mehrere Regeln zutreffen?

Welche Farbe bekommt diese Überschrift?

```html
<div class="main-content">
  <h1 id="main-title">Überschrift</h1>
</div>
```

```css
.main-content h1 {
  color: blue;
}

h1#main-title {
  color: green;
}

h1 {
  color: red;
}
```

Definition

Definition

Sollten mehrere CSS-Regeln im Widerspruch zueinander stehen, wird die anzuwendende Regel wie folgt ermittelt:

1. **Wichtigkeit (!important,...)**
2. **Spezifität**
3. **Platzierung (im Quellcode)**

Berechnung der Wichtigkeit

1. Standardwerte vom Browser („User Agent Stylesheet“)
2. CSS-Deklarationen der Webseite
3. CSS-Deklarationen der Webseite (mit !important)

Aber was passiert, wenn die Wichtigkeit identisch ist?

=> Dann wird auf Basis der „Spezifität“ entschieden!

Fortgeschrittenes CSS und SASS

Wie wird CSS verarbeitet Teil 2: Spezifität

Jannis Seemann

Berechnung der Spezifität

1. Inline Styles
2. Anzahl der IDs
3. Anzahl der Klassen-, Pseudoklassen- und Attribute
4. Elemente, Pseudo-Elemente

```
.main-content h1 {
  color: blue;
}
```

=> Spezifität: (0, 0, 1, 1)

```
h1#main-title {
  color: green;
}
```

=> Spezifität: (0, 1, 0, 1)

=> Die Überschrift wird grün!

```
div h1 {
  color: red;
}
```

=> Spezifität: (0, 0, 0, 2)

Was passiert, auch noch die Spezifität identisch ist?

Definition: Dann wird die Reihenfolge im Quellcode betrachtet.

```
.main-content h1 {
  color: blue;
}

.main-content h1 {
  color: magenta;
}
```

=> Hier wird die Überschrift in magenta angezeigt

Was folgt daraus für dich in der Praxis?

- **!important:**
  - Diese Deklarationen haben die höchste Wichtigkeit
  - Dadurch wird es aber schwer, wiederverwendbaren CSS-Code zu schreiben
  - Versuche also, !important zu vermeiden
- **Inline-Styles (<h1 style="color: red;">Überschrift</h1>)**
  - Diese sind spezifischer als Deklarationen aus externen CSS-Dateien
  - Sorgen aber auch dafür, dass der Code schwerer zu warten wird
- **Best-Practice:**
  - Innerhalb unserem CSS-Code arbeiten wir mit der Spezifität von CSS-Selektoren
  - Externen CSS-Code (z.B. Bootstrap,...) binden wir aber weiterhin vor unserem Anwendungs-Code ein

Fortgeschrittenes CSS und SASS

Wie werden CSS-Eigenschaften vererbt?

Jannis Seemann

CSS-Eigenschaften, Definitionen

- Jede CSS-Eigenschaft existiert für jedes Element und hat einen initialen Wert
- Diesen können wir über den Wert "initial" abrufen
- Zudem: Manche Eigenschaften werden vererbt
- **Wichtig:**
  - Diese Werte werden vor dem Vererben zuerst umgerechnet!
  - Prozentwerte bzw. relative Werte werden in Pixel umgerechnet
  - Erst dann findet die Vererbung statt
- => Anschließend können die Elemente dann vom Browser angezeigt werden

Fortgeschrittenes CSS und SASS

Wie wird die Schriftgröße definiert?

Wie können wir Schriftgrößen angeben?

- **Möglichkeit 1:**
  - Die Schriftgröße wird direkt in Pixeln angegeben
- **Möglichkeit 2:**
  - Die Schriftgröße wird in em angegeben
  - Die Schriftgröße wird relativ zur Schriftgröße des Parent-Elementes berechnet
- **Möglichkeit 3:**
  - Die Schriftgröße wird in rem angegeben
  - Die Schriftgröße ist dann relativ zur Schriftgröße des HTML-Elementes

Fortgeschrittenes CSS und SASS

Wie wird die Breite bzw. Höhe von einem Element berechnet?

Jannis Seemann

Inline vs. Block

- **Bei einem Inline-Element:**
  - Dieses ist mit im Textfluss (inline)
  - Die Größe dieses Elementes ergibt sich automatisch aus dem Inhalt
  - Es kann daher weder eine Breite, noch eine Höhe definiert werden
  - Padding bzw. Margin ist nur in der horizontalen (rechts bzw. links) erlaubt
- **Ein Inline-Block-Element:**
  - Ein Mix aus Block- und Inline-Element
  - Standardmäßig so groß wie ein Inline-Element
  - Aber wir dürfen eine Größe definieren
- **Block-Element:**
  - Standardmäßig eine Breite von 100%
  - Nur die Höhe ergibt sich aus dem Inhalt des Elementes

Fortgeschrittenes CSS und SASS

Wie wird die Breite bzw. Höhe von einem Element definiert?

Jannis Seemann

Größenangaben von Elementen

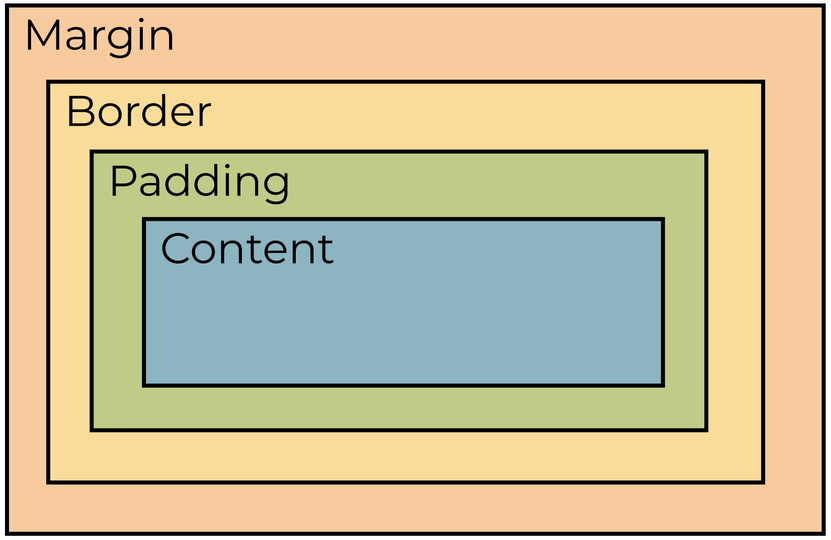
- **Wir können hier verschieden Einheiten benutzen:**
  - **px:** Hierbei wird die Größe direkt angegeben
  - **%:** Hierbei bezieht sich die Größe auf die Größe des Parent-Elementes
  - **em bzw. rem:** Hierbei bezieht sich die Größe auf die Schriftgröße des aktuellen Elementes (em) bzw. <html>-Elementes (rem)
  - **vw & vh:** Hierbei bezieht sich die Größe auf die Größe des Viewports des Browsers

Fortgeschrittenes CSS und SASS

Box-Sizing: content-box vs. border-box

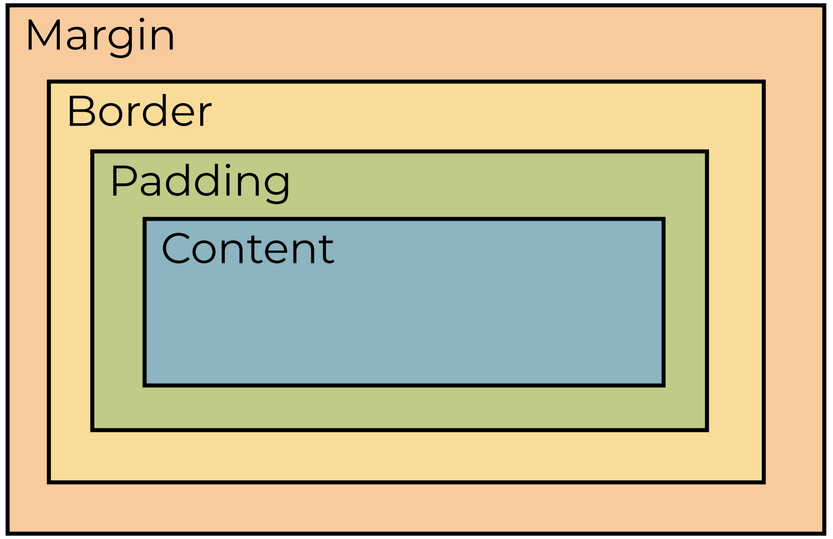
Jannis Seemann

Padding, Margin & Border



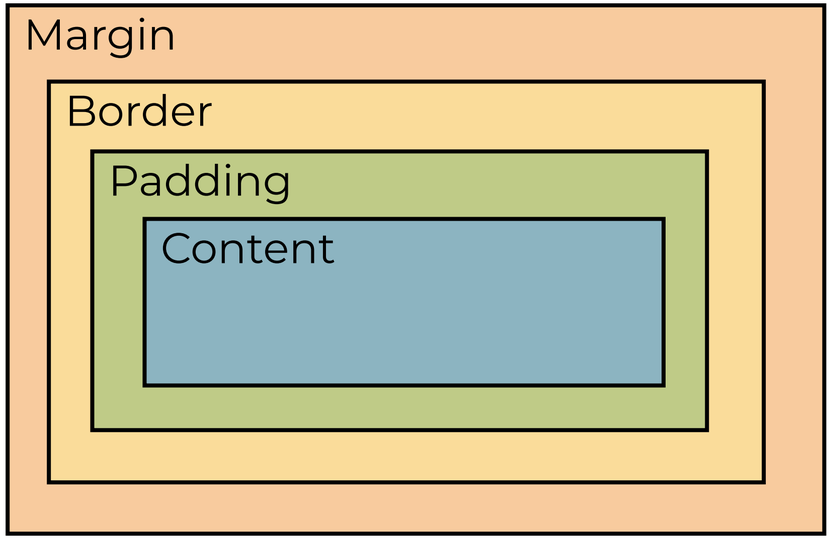
- Ein Block-Element hat verschiedene „Bereiche“
- Für diese Bereiche gibt es unterschiedliche Befehle, um die jeweilige Größe zu steuern
- **Standardmäßig gilt hierbei:**
  - Width und height: Steuert die Größe des Inhaltsbereiches
  - Padding: Steuert den Innenabstand
  - Border: Rahmen vom Element
  - Margin: Steuert den Außenabstand

Padding, Margin & Border



- **Beispielrechnung:**
  - Ein Element der Breite (width) von 200px
  - Mit einem Padding von 10px an jeder Seite
  - Mit einer Border von 5px an jeder Seite
  - Dieses Element ist dann effektiv 200px + 2 * 10px + 2 * 5px = 230px breit
- **Warum / Wann ist dies ein Problem?**
  - Kindelement mit Breite von 100%
  - Dazu kommt noch eine Border von 3px
  - Dieses Element hat eine effektive Breite von 100% + 6px
  - Es ragt also über den Inhaltsbereich vom Elternelement hinaus!

Padding, Margin & Border



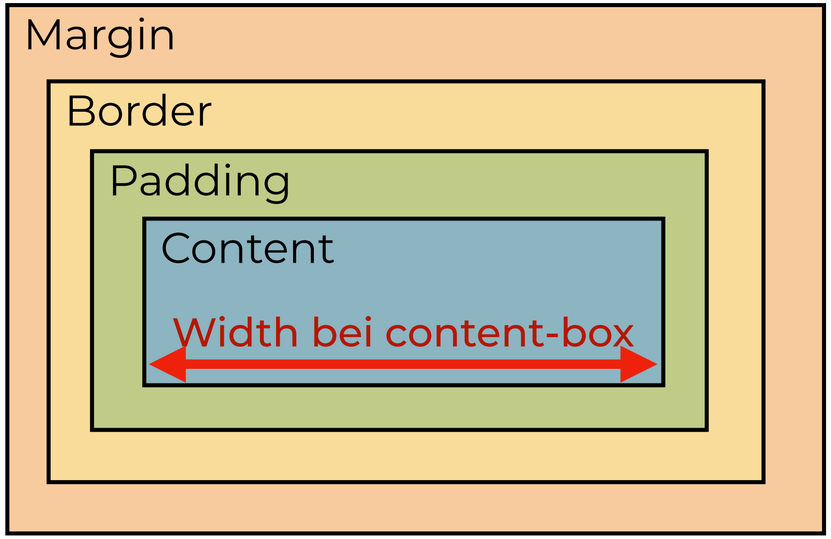
- **Genau dieses Problem löst das Box-Sizing:**
  - **CSS-Regel:**

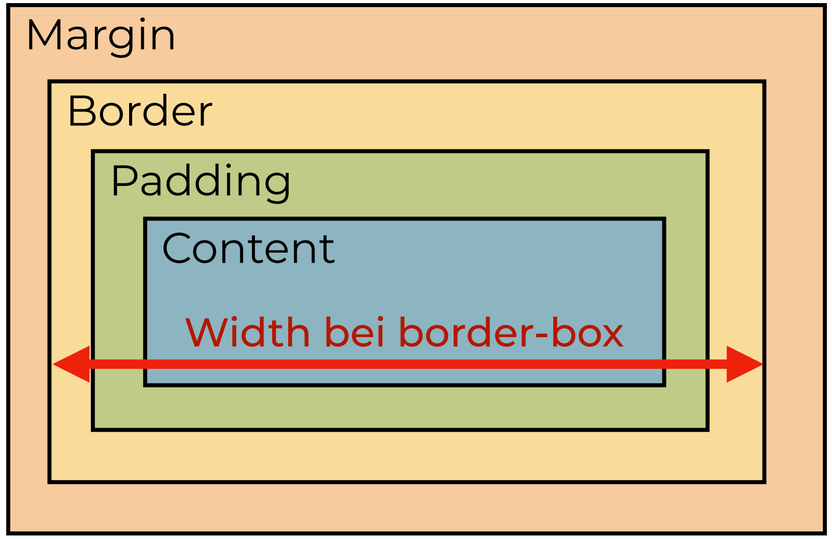
    ```
    * { box-sizing: border-box; }
    ```

  - Hiermit wird für alle Elemente (Selektor: *) die Berechnung der Breite abgeändert
  - Die Befehle width bzw. height beziehen sich jetzt auf die Größe vom Content + Padding + Border!
- **Beispiel:**
  - Element mit Breite von 100%, box-sizing ist border-box
  - Border von 3px
  - Dieses Element hat eine effektive Breite (mit Border) von 100%
  - Der Rahmen reduziert die Größe vom Inhaltsbereich
  - Es ragt also nicht über die Breite vom Parent hinaus

content-box vs. border-box

- **Analog gilt dies natürlich auch für die height-Eigenschaft in CSS!**





Box-Sizing, Best Practices

- **Best-Practice:**
  - **Wir verwenden für alle Elemente border-box als box-sizing:**

    ```
    * { box-sizing: border-box; }
    ```

  - Wenn wir für bestimmte Elemente dann doch mal die andere Rechenart benötigen, können wir dies ja immer noch ändern
- **Interessante Info:** Die CSS-Working-Group sagt selbst, dass sie das Box-Sizing standardmäßig auf border-box hätten setzen sollen:
  - Dieser Fehler kann aber nicht mehr „glattgebügelt“ werden
  - **Das Problem:** Moderne Browser würden dann ja ältere Webseiten nicht mehr korrekt anzeigen
  - Daher bleibt dies im CSS-Standard weiterhin enthalten
  - Und wir müssen es manuell auf border-box setzen
  - Link zur Quelle: https://wiki.csswg.org/ideas/mistakes

Fortgeschrittenes CSS und SASS

Idee: Responsive Design

Idee: Responsive Design

- **Grundidee:**
  - Wir möchten mit einer Seite alle Geräteklassen abdecken
  - Handys, Tablets, Computer
- **Wir haben hier 2 Möglichkeiten:**
  - **Mobile first:**
    - Wir entwickeln zuerst für Mobilgeräte
    - **Ergebnis:** Seite ist abgespeckt, auf die Hauptfunktionen
  - **Desktop first:**
    - Wir entwickeln zuerst für Desktop
    - Und passen dann die Seite für Mobilgeräte an
    - **Ergebnis:** Seite kann auf Mobilgeräten leicht überladen wirken
- **Tendenziell gilt:**
  - Mobile first entwickeln
  - Oder Desktop-first, und gleichzeitig über Mobilversion nachdenken

Idee: Responsive Design, Desktop first

- **Desktop first:**
  - Wir setzen zuerst die Eigenschaften für den Desktop
  - Und passen dann die Seite an
  - **Ergebnis:**

```
body {
  background-color: white;
}

@media (max-width: 400px) {
  body {
    background-color: green;
  }
}
```

Idee: Responsive Design, Mobile first

- **Mobile first:**
  - Wir setzen zuerst die Eigenschaften für Mobilgeräte
  - Und passen dann die Seite an
  - **Ergebnis:**

```
body {
  background-color: green;
}

@media (min-width: 400px) {
  body {
    background-color: white;
  }
}
```

Fortgeschrittenes CSS und SASS

SVG und HTML & CSS

Jannis Seemann

SVG, HTML & CSS

- HTML baut auf XML auf
- Genauso das Bildformat SVG
- Wir können daher ein SVG-Bild mit in unserem HTML-Code verwenden:
- **Beispiel:**

```
<div class="svg-image">
    <svg width="100" height="100">
        <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
    </svg>
</div>
```

SVG in HTML

```
<div class="svg-image">
    <svg width="100" height="100">
        <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
    </svg>
</div>
```

- **Wichtig hierbei:**
  - SVG-Elemente sind XML-Elemente, aber keine HTML-Elemente!
  - Das bedeutet konkret:
    - CSS funktioniert hier ähnlich, aber etwas anders ;)

Fortgeschrittenes CSS und SASS

Text und Bilder vertikal ausrichten (vertical-align)

Jannis Seemann

Vertikale Ausrichtung von Text & Bildern

- Über vertical-align können wir steuern, wie sich ein Element auf Basis der Grundlinie des Parent-Elementes ausrichtet
- Die wichtigsten Optionen sind:
  - Baseline: Das Objekt liegt auf der Grundlinie vom Parent-Elementes
  - text-top: Die Oberkante vom Objekt liegt auf der Oberkante des Textes des Parent
  - text-bottom: Die Unterkante vom Objekt liegt auf der Unterkante des Textes des Parent
  - middle: Die Mitte vom Objekt liegt auf der Mittellinie der Kleinbuchstaben vom Parent
- Das sind die wichtigsten Optionen...
- Für weitere Optionen möchte ich dich auf die Dokumentation verweisen, oder z.B. die MDN Web Docs
  - => https://developer.mozilla.org/de/docs/Web/CSS/vertical-align

Fortgeschrittenes CSS und SASS

Der Header-Bereich: Was erwartet dich?

Jannis Seemann

Der Header-Bereich

- **Was erwartet dich?**
  - **Block Element Modifier: Schreibe wiederverwendbaren CSS-Code**
  - **Wir setzen den Header-Bereich der Seite um!**

Fortgeschrittenes CSS und SASS

CSS-Code strukturieren, Block Element Modifier

Jannis Seemann

Idee: CSS-Code strukturieren

- **Grundidee:**
  - Können wir IDs für CSS-Code verwenden?
  - Jede ID darf maximal 1x auf einer Seite vorkommen
  - Das kann in der Praxis schwierig sein
  - Wir setzen daher primär auf Klassen, um Elemente anzusteuern
- **Wie vergeben wir die Klassennamen?**
- **Ziel:**
  - **Wiederverwendbarer CSS-Code**
  - **Wichtig:** Aber so, dass wir auch noch Veränderungen vornehmen können
  - **Beispiel:**
    - Button, Standardmäßige Farbe: Grau
    - Wir möchten aber auch einen Button z.B. in Blau definieren
    - Dieser soll vom CSS-Code auf dem grauen Button aufbauen

Idee: Block Element Modifier

- **Grundidee:**
  - Wir strukturieren unsere Seite in verschiedene „Blöcke“
  - In diesen gibt es dann Elemente, diese werden über einen „__“ gekennzeichnet
  - All diese Elemente können über 2 Bindestriche abgeändert werden
  - Wichtig: Dies ist nur eine Möglichkeit, an die wir uns halten können - BEM macht aber in der Praxis oft Sinn!
- **Beispiel:**

```
<form class="form form--theme-xmas form--simple">
  <input class="form__input" type="text" />
  <input
    class="form__submit form__submit--disabled"
    type="submit" />
</form>
```

Idee: Block Element Modifier

- **Weitere Regeln:**
  - **Wir versuchen komplexe Selektoren zu vermeiden:**
    - Wir vermeiden als Selektor:
    - .form input[type="submit"]
  - **Und vergeben in diesem Fall dann lieber eine neue Klasse:**
    - <input class="form__submit" ... />

Fortgeschrittenes CSS und SASS

Header, Positionierung: Was erwartet dich?

Jannis Seemann

Header, Positionierung

- **Was erwartet dich?**
  - Wie positionierst du Elemente?
  - Wir nutzen dies, um die Berge über der Schrift zu platzieren
  - Responsive Design!

Fortgeschrittenes CSS und SASS

Wie können wir Elemente positionieren?

Jannis Seemann

Positionierung von Elementen

- **position: relative**
  - Hierbei wird das Element relativ zu seiner normalen Position positioniert
  - Anderer Inhalt wird nicht angepasst, um die Lücke dieses Elementes aufzufüllen
- **position: fixed**
  - Hierbei wird das Element relativ zum Viewport positioniert
  - Es bleibt also dort, selbst wenn die Seite gescrollt wird
- **position: sticky**
  - Das Element wechselt zwischen relative und fixed, je nachdem, wie der Benutzer gerade gescrollt hat
  -

Positionierung von Elementen

- **position: absolute**
  - Hierbei wird das Element aus dem Textfluss „herausgehoben“
  - Es wird dann relativ zum nächsten Eltern-Element mit Positionierung positioniert
  - Wenn es keins gibt, dann wird es absolut auf der gesamten Seite positioniert
  - Das sollten wir uns mal genauer anschauen im Beispiel!

Position vs. Translate?

- Es gibt jetzt aber auch noch die transform-Eigenschaft
- Damit können wir Elemente auch positionieren
- **Hierbei gilt:**
  - **Position:**
    - Dies sollten wir normalerweise verwenden
  - **Transform:**
    - Wenn wir die Position allerdings animieren möchten, sollten wir transform verwenden
    - Der Browser kann ein transform leichter auf die GPU auslagern

```css
h1 {
  position: relative;
  top: -50px;
}
```

```css
h1 {
  transform: translateY(-50px);
}
```

Z-index

- **Z-index:**

  - Wenn ein Element positioniert ist (position: relative oder position: absolute)
  - Können wir die Zeichenreihenfolge über die Eigenschaft z-index beeinflussen
  - Ein niedriger z-index liegt dann tiefer als ein höherer z-index

Fortgeschrittenes CSS und SASS

Wie können wir Elemente nebeneinander positionieren?

Jannis Seemann

Elemente nebeneinander positionieren

- **Hierzu gibt es verschiedene Möglichkeiten**
  - Die einfachste ist mit einem float: left bzw. right
  - Andere Möglichkeiten: Flexbox bzw. CSS-Grid - dazu später mehr!
- Wenn wir ein **float: left** auf ein Block-Element anwenden:
  - Wird dieses Element aus dem normalen Elementfluss herausgenommen
  - „Fließen“ andere Elemente um dieses Element herum
  - Die Breite ergibt sich auf Basis des Inhaltes
  - Wächst das Eltern-Element nicht in der Höhe
- **Um das Float zu beenden:**
  - Können wir dem Eltern-Element ein clear: both geben

Fortgeschrittenes CSS und SASS

Schriftarten einbinden

Jannis Seemann

Schriftarten einbinden

- **Für das Einbinden von Schriftarten gibt es verschiedene Formate:**
  - SVG
  - EOT (Embedded Open Type) - Internet Explorer älter gleich Version 8.0
  - OTF / TTF (OpenType Font / TrueType Font) - Vorläufer vom WOFF
  - WOFF: Web Open Font Format
  - WOFF2: Web Open Font Format (aber bessere Komprimierung)

Schriftarten einbinden

- **Früher war es so:**
  - Für jede Schriftdicke wird eine komplett neue Schriftdicke geladen
  - Inzwischen gibt es "Variable Fonts"
  - Bei Schriften, die dies unterstützen, kann u.U. die Schriftdicke angepasst werden
  - Wir können also mit einer Datei alle Schriftdicken abdecken!
- **Wie binden wir Schriftarten ein?**
  - Beispielsweise über Google Web Fonts
    - **Datenschutz???**
- **Wir können die Schriftart auch direkt bei uns platzieren**

Fortgeschrittenes CSS und SASS

Schriftarten: Was erwartet dich?

Schriftarten einbinden

- **Was erwartet dich?**
  - Schriften lokal ausliefern
  - Welche Dateiformate benötigen wir?
  - Wie funktioniert der Unicode-Range?
  - Schriften mit variabler Schriftdicke, Animation

Fortgeschrittenes CSS und SASS

CSS-Workflow: SASS / SCSS

Jannis Seemann

Fortgeschrittenes CSS und SASS

SASS / SCSS: Was erwartet dich?

SASS / SCSS: Was erwartet dich?

- Ein Tool um übersichtlicheren CSS-Code zu schreiben
- Wird die Grundlage sein, dass wir unseren CSS-Code auf verschiedene Dateien aufteilen können
- Darauf wird dann bald auch unser Photography-Projekt aufbauen

Fortgeschrittenes CSS und SASS

Vendor-Prefix

Vendor-Prefix

- **Neue Features werden hinter einem Vendor-Prefix entwickelt**
- **Beispiel:**
  - -webkit-animation
  - Der Syntax vom Wert (auf der rechten Seite vom Doppelpunkt) wird dann teilweise noch geändert
  - Jeder Browserhersteller hat hier i.d.R. sein eigenes Prefix
- **Wenn das Feature dann fertig ist:**
  - Wird der Syntax nicht mehr geändert
  - Wird das Feature übernommen
  - Es ist dann irgendwann Prefix verfügbar:
    - **Hier:** animation

Vendor-Prefix

- Was bedeutet das für uns?
- Teilweise können wir neue Features vorab verwenden, wenn wir das Vendor-Prefix mit verwenden
- **Beispiel:**

```
-webkit-animation-name: header__heading-main-font-weight;
        animation-name: header__heading-main-font-weight;
-webkit-animation-delay: 0.75s;
        animation-delay: 0.75s;
-webkit-animation-duration: 2s;
        animation-duration: 2s;
-webkit-animation-timing-function: ease-in-out;
        animation-timing-function: ease-in-out;
-webkit-animation-fill-mode: both;
        animation-fill-mode: both;
```

Vendor-Prefix

- Was für Präfixe gibt es?
- **Primär sind relevant:**
  - -webkit-:
    - Alle Browser, die auf der Webkit-Engine aufbauen:
    - Chrome
    - Safari
    - Neuere Versionen von Opera
    - Neuere Versionen von Edge
    - Firefox für iOS
  - -moz-:
    - Firefox auf Desktop-Geräten
  - -ms-:
    - Internet Explorer
    - Ältere Versionen von Microsoft Edge

Vendor-Prefix: Autoprefixer

- Gerade vor ein paar Jahren war dies extrem wichtig
- Heutzutage werden zwar fast alle relevanten Features auch ohne Vendor-Prefix unterstützt
- **Dennoch können wir einen Schritt durchführen:**
  - Wir lassen die relevanten Präfixe automatisch hinzufügen
  - Wir müssen uns um nichts kümmern
  - Und bekommen bessere Browserunterstützung!

Fortgeschrittenes CSS und SASS

SASS / SCSS

Jannis Seemann

Motivation 1

- **Bisher haben wir bei unserem CSS-Code ein grundsätzliches Problem:**
  - Der CSS-Code wird recht lang und dadurch unübersichtlich
  - Können wir den CSS-Code nicht auf verschiedene Dateien aufteilen?
  - Klar, aber dann muss der Browser mehrere Dateien einbinden
  - Das kann zu längeren Ladezeiten führen

```
<link href="./styles/main-all.css" type="text/css" rel="stylesheet">
<link href="./styles/main-header.css" type="text/css" rel="stylesheet">
<link href="./styles/main-naviation.css" type="text/css" rel="stylesheet">
```

Motivation 2

- **Teils ist unser CSS-Code recht unübersichtlich**
- **Konkret:**
  - Wir können Regeln nicht verschachteln
  - Media-Queries müssen außerhalb von den Selektoren stehen
  - Wir können keine wiederverwendbaren "Funktionen" schreiben

Idee: SASS / SCSS

- Wir schreiben SCSS statt CSS
- Und das wird für uns in normales CSS umgewandelt
- Das normale CSS kann dann vom Browser verstanden werden
- Wir haben also einen Zwischenschritt:

```scss
.header__heading-main {
    font-size: 6rem;
    @media (max-width: 62em) {
        font-size: 5rem;
    }
}
```

SASS / SCSS

```css
.header__header-main {
    font-size: 6rem;
}
@media (max-width: 62em) {
    .header__heading-main {
        font-size: 5rem;
    }
}
```

SCSS:

```scss
.header__heading-main {
    font-size: 6rem;
    @media (max-width: 62em) {
        font-size: 5rem;
    }
}
```

SASS:

```sass
.header__heading-main
    font-size: 6rem
    @media (max-width: 62em)
        font-size: 5rem
```

Fortgeschrittenes CSS und SASS

SCSS nach CSS umwandeln

SCSS nach CSS umwandeln

- Normalerweise gäbe es hier wunderschöne Erweiterungen direkt für Visual Studio Code
- Diese bauen aber auf node-sass auf
- node-sass baut auf LibSass auf
- LibSass und auch node-sass sind aber deprecated und unterstützt ein paar SCSS-Features nicht (@use)
- Viele Erweiterungen nutzen noch die ältere Implementierung über node-sass nach LibSass
- => Wir müssen uns unser eigenes Setup bauen!

SCSS nach CSS umwandeln

- Wie sieht unser Setup aus?

Fortgeschrittenes CSS und SASS

Wie können wir unseren SCSS-Code strukturieren?

Jannis Seemann

Vorgeschlagene Projektstruktur

- **/abstracts:**
  - Hier platzieren wir Mixins, Funktionen und Platzhalter
  - Schauen wir uns später noch genauer an!
- **/base**
  - Hier werden grundlegende Styles definiert
  - z.B. Schriftarten, CSS-Reset, ...
- **/components**
  - Hier werden kleinere Komponenten definiert (z.B. Buttons, Formulare,...)
- **/layout**
  - Hier wird das allgemeine Layout definiert

Vorgeschlagene Projektstruktur

- **/pages**
  - Hier wird Code für einzelne Seiten platziert
- **/themes**
  - Hier können wir Code definieren, der für verschiedene Themes benötigt wird.
  - Dies wird nicht für jedes Projekt benötigt
- **/vendors**
  - Hier platzieren wir Code, der CSS-Code von anderen Projekten lädt
  - Beispielsweise wenn wir Bootstrap einbinden möchten:
    - Platzieren wir hier eine .scss-Datei, die dann Bootstrap einbindet
- **main.scss:**
  - Von hier aus wird alles geladen

Fortgeschrittenes CSS und SASS

SASS: Expertenwissen
Was erwartet dich?

Jannis Seemann

SASS: Expertenwissen

- Was erwartet dich?
  - Variablen in SASS
  - Der Scope der Variablen
  - Farben mit SASS abändern
  - Variablen in Konfigurations-Datei auslagern
  - Expertenwissen zu SASS-Modulen

Fortgeschrittenes CSS und SASS

SASS: Variablen

Jannis Seemann

SASS: Variablen

- In SASS bzw. SCSS können wir Variablen definieren
- Das geht über die $-Schreibweise
- **Beispiel:**
  - $color-dark: black;
- Diese Variable können wir dann später verwenden:
  - h1 { color: $color-black; }
- **Wichtig:**
  - In CSS selbst gibt es auch Variablen
  - Das haben wir uns noch nicht angeschaut
  - Diese funktionieren aber komplett anders, als die Variablen in SASS
  - Diese schauen wir uns später nochmal genauer an!

Fortgeschrittenes CSS und SASS

SASS: Variablen (Scope)

Jannis Seemann

SASS: Scope von Variablen

- **Variablen, die wir in einem CSS-Deklaration-Bereich erstellen:**
  - Sind lokale Variablen
  - Stehen uns nur in diesem Bereich zur Verfügung
- **Variablen, die wir außerhalb erstellen:**
  - Stehen uns für den gesamten Rest der aktuellen Datei zur Verfügung

Fortgeschrittenes CSS und SASS

SASS: Farben

Jannis Seemann

SASS: Farben

- **SASS kann für uns Farben abändern**
  - Dafür steht uns das Modul sass:color zur Verfügung
  - Dieses können wir per @use laden
  - Und dann in unserer Anwendung nutzen!
  - Dazu schauen wir uns aber zuerst mal die Dokumentation an...

Fortgeschrittenes CSS und SASS

SASS: @extend

Jannis Seemann

SASS: @extend

- Mit @extend können wir einen existierenden Selektor erweitern
- **Beispielsweise könnten wir 2 Klassen haben:**
  - button-green
  - button-blue
- Für beide Buttons können wir dann gemeinsame Styles definieren...
- **Wichtig:**
  - Wenn wir uns in einem Projekt an die BEM-Schreibweise halten, benötigen wir i.d.R. kein @extend!
  - Das sollten wir uns im Video aber nochmal genauer anschauen...

Fortgeschrittenes CSS und SASS

SASS: Variablen und @use

Jannis Seemann

SASS: Variablen und @use

- Bisher haben wir Variablen immer nur in der selben Datei verwendet
- Aber oft möchten wir eine zentrale Konfiguration haben
- Was können wir zentral definieren?
  - Breakpoints
  - Farben für unsere Anwendung
  - Abstandswerte
- Aber dafür müssen wir Variablen aus anderen Dateien verwenden können
- Hierbei ist es, so dass über @use ein neuer Namespace erstellt wird
- Und auf diesen können wir von außen zugreifen

Fortgeschrittenes CSS und SASS

SASS: Best-Practices mit Variablen

Jannis Seemann

SASS: Variablen, Best-Practice

- Globale Variablen definieren wir am Anfang der Datei
  - Anschließend hängen wir an die Variablendefinition ein !default
  - **Beispiel:**
    - `$color: black!default;`
- Zudem:
  - Alle Variablen definieren wir in einem Ordner
  - Und laden dann die

Fortgeschrittenes CSS und SASS

CSS: Die filter-Eigenschaft

Jannis Seemann

CSS: Die filter-Eigenschaft

- Mit dem filter können wir einen grafischen Filter auf ein Element anwenden
- Hierzu wird zuerst das Element selbst angezeigt
- Und anschließend wird der Filter auf dieses Element angewendet
- **Welche Filter-Möglichkeiten gibt es?**

```
filter: blur(5px);
filter: brightness(0.4);
filter: contrast(200%);
filter: drop-shadow(16px 16px 20px blue);
filter: grayscale(50%);
filter: hue-rotate(90deg);
filter: invert(75%);
filter: opacity(25%);
filter: saturate(30%);
filter: sepia(60%);
```

Fortgeschrittenes CSS und SASS

Motivations-Bereich: Was erwartet dich?

Motivations-Bereich: Was erwartet dich?

- **Was erwartet dich?:**
  - Farben von Bildern verändern: Die Filter-Eigenschaft
  - Schicker Hover-Effekt

Fortgeschrittenes CSS und SASS

CSS: Layouts mit Flexbox

Jannis Seemann

Fortgeschrittenes CSS und SASS

Flexbox: Was erwartet dich in diesem Abschnitt?

Flexbox: Was erwartet dich?

- **Flexbox:**
  - Flexible und effiziente CSS-Layouts
  - Und, endlich in CSS möglich: Element sowohl vertikal, als auch horizontal zentrieren

Fortgeschrittenes CSS und SASS

CSS: Warum benötigen wir Flexbox?

Jannis Seemann

CSS: Layout-Modelle

- Bisher haben wir uns schon 2 verschiedene Layout-Modelle angeschaut:
  - **Block-Modell:**
    - Elemente werden als Block untereinander angezeigt
  - **Inline-Modell:**
    - Elemente werden in einer Textzeile nebeneinander angezeigt
- Die Praxis hat aber gezeigt, dass diese Modelle nicht immer ausreichen
- **Beispiel:**
  - Eine Element soll vertikal zentriert werden
  - Die Anwendung besteht aus verschiedenen Spalten
    - Diese sollen auf Mobilgeräten in einer anderen Reihenfolge angezeigt werden
  - **Zudem:** Die Positionierung per "float" hat unerwünschte Nebeneffekte, wenn wir es für ein Layout verwenden

Fortgeschrittenes CSS und SASS

Flexbox: Wie funktioniert es?

Flexbox: Wie funktioniert es?

- **Flexbox-Container:**
  - Das ist das Parent-Element, welches die einzelnen Spalten enthält
  - Dieses Element bekommt ein "display: flex;"
  - Hier können wir dann spezielle CSS-Eigenschaften verwenden, um die Flexbox zu steuern:
  - **Insbesondere sind für uns wichtig:**
    - flex-direction
    - justify-content
    - align-items
  - Das sollten wir uns aber anhand eines Beispiels genauer anschauen!

Fortgeschrittenes CSS und SASS

Flexbox: Einzelnes Element steuern

Jannis Seemann

Flexbox: Einzelnes Element steuern

- **Flexbox-Element:**
  - Ein Element wird dadurch zu einem Flexbox-Element, indem das Parent-Element ein Flexbox-Container ist
- **Dann stehen uns weitere Befehle zur Verfügung:**
  - **align-self**
  - **order**
  - **flex:**
    - flex-basis
    - flex-shrink
    - flex-grow

Fortgeschrittenes CSS und SASS

Der Tours-Bereich

Jannis Seemann

Der Tours-Bereich

- In diesem Abschnitt setzen wir den Touren-Bereich um
- **Was erwartet dich?**
  - Eine gute Anwendung für eine Flexbox
  - **Schicke Animation:**
    - Sehr effizient: Animation erfolgt per transform
    - Wir lösen ein komplexes z-index-Problem
  - Schrift-Outline anzeigen

Fortgeschrittenes CSS und SASS

Story-Bereich

Story-Bereich

- In diesem Abschnitt setzen wir den Angebots-Bereich um
- **Was erwartet dich?**
  - Hintergrund-Video
  - Die Größe vom Hintergrund-Video steuern
  - Schicker Transparenzeffekt im Chrome
  - Text automatisch in mehreren Spalten anzeigen lassen

Fortgeschrittenes CSS und SASS

Offers-Bereich

Jannis Seemann

Offers-Bereich

- In diesem Abschnitt setzen wir den Angebots-Bereich um
- **Was erwartet dich?**
  - Elemente mit Flexbox layouten
  - Komplexe Farbeffekte mit CSS (mix-blend-mode)
  - Elemente pixelgenau platzieren

Fortgeschrittenes CSS und SASS

3D in CSS

Jannis Seemann

3D in CSS

- **Bisher:**
  - Unsere Webseite war komplett 2D
  - Selbst Befehle wie transform: scale(1.2) haben einfach nur ein Element vergrößert
- **In diesem Abschnitt:**
  - Wirst du lernen, wie du echte 3D-Effekte erzeugen kannst

Fortgeschrittenes CSS und SASS

3D in CSS

Jannis Seemann

3D in CSS

- **CSS unterstützt 3D-Effekte:**
  - Diese können wir wie gewohnt über die transform-Eigenschaft setzen
  - Hier gibt es jetzt aber neue Befehle:
    - translate3d bzw. translateX, translateY, translateZ
    - rotate3d bzw. rotateX, rotateY, rotateZ
    - scale3d bzw. scaleX, scaleY, scaleZ
    - matrix3d
  - **Diese sollten wir uns mal im Code genauer anschauen**

Fortgeschrittenes CSS und SASS

CSS-Variablen

Jannis Seemann

CSS-Variablen

- **Was erwartet dich in diesem Abschnitt?**
  - Komplett unterschiedlich zu SASS-Variablen
  - Wir können diese verwenden, um komplexere Designs umzusetzen
- **Bei uns:**
  - Wir werden es nutzen, um die aktuelle Mausposition in einem Element von JavaScript nach CSS zu übertragen
  - Und den eigentlichen "Effekt" können wir dann komplett in CSS entwickeln

Fortgeschrittenes CSS und SASS

CSS-Variablen

Jannis Seemann

CSS-Variablen

- **CSS-Variablen können wir über folgende Schreibweise setzen:**
  - `body { --color-main: red; }`
- **Dann können wir sie in einem Element benutzen. Der 2. Parameter definiert hierbei den Standardwert:**
  - `h1 { color: var(--color-main, green); }`
- **Wichtig: CSS-Variablen:**
  - Diese Variablen werden wie CSS-Eigenschaften vererbt, und gelten für alle Kind-Elemente im DOM-Tree
  - Der Browser wertet also die Variable aus!
- **Im Gegensatz dazu, SCSS-Variablen:**
  - Werden zur Kompilier-Zeit von SCSS gesetzt, ausgewertet und eingesetzt
  - Davon bekommt der Browser nichts mit!

Fortgeschrittenes CSS und SASS

CSS-Variablen: Wann verwenden?

CSS-Variablen vs. SASS-Variablen

- **SASS-Variablen:**
  - Oft sehr viel übersichtlicher
  - Werden vom Compiler verarbeitet, das Ergebnis ist klar, und steht im CSS-Code drinnen
- **CSS-Variablen:**
  - Können überschrieben werden, ohne dass wir es wissen
  - Und werden dann u.U. komisch angezeigt
- **Zudem: Wie ist der Browser support?**
  - => https://caniuse.com/?search=css%20variable

CSS-Variablen vs. SASS-Variablen

- **SASS-Projekt:**
  - Wir verwenden primär SASS-Variablen
  - Und verwenden CSS-Variablen nur, wenn sie unseren SASS-Code signifikant vereinfachen
- **nicht-SASS-Projekt:**
  - Hier können wir dann schon CSS-Variablen verwenden
  - Wenn wir alle Variablen primär für den body deklarieren, gibt es auch keine Probleme, dass Variablen überschrieben werden
- **Zudem:**
  - CSS-Variablen können uns erlauben, Module von anderen Entwicklern zu stylen!

Fortgeschrittenes CSS und SASS

CSS-Grid

Jannis Seemann

CSS-Grid

- CSS Grid ist eine quasi die 2D-Version einer Flexbox
- Damit können wir noch komplexere Layouts designen
- **Und:**
  - Das ist die Grundlage für den nächsten Bereich von unserem Projekt

Fortgeschrittenes CSS und SASS

CSS-Grid

Jannis Seemann

CSS-Grid

- **Wir haben jetzt schon diverse Layout-Möglichkeiten in CSS kennengelernt:**
  - Inline
  - Block
  - Flexbox
- Warum benötigen wir jetzt noch eine?
- **Idee:**
  - Flexbox ist für eindimensionale Layouts (eine Spalte bzw. eine Zeile)
  - Grid ist für mehrdimensionale Layouts (mehrere Spalten und mehrere Zeilen)
- Aber wie funktioniert Grid?

Fortgeschrittenes CSS und SASS

Der Gallery-Bereich

Jannis Seemann

Der Gallery-Bereich

- **In diesem Bereich:**
  - Eine Anwendung von CSS-Grid
  - Ein Bild wird hierbei größer angezeigt als die anderen Bilder
- **Zudem:**
  - Der Checkbox-Trick: Interaktive Elemente, komplett per CSS
  - Bilder mit "Instagram-Filter" versehen: Komplett per CSS

Fortgeschrittenes CSS und SASS

Der Über-uns-Bereich

Jannis Seemann

Der Über-uns-Bereich

- **In diesem Bereich:**
  - Bild wird von Text umflossen
  - Das Bild ist hierbei rund, nicht eckig!
  - Hierzu gibt es Spezial-Befehle in CSS, um dies zu realisieren
- **Zudem:**
  - Statt nur einem Bild gibt es eine Slideshow!
  - Diese funktioniert komplett per CSS
  - Das ist eine weitere Anwendung von mehreren @keyframes-Animationen

Fortgeschrittenes CSS und SASS

CSS: Target abfragen / Modal implementieren

Jannis Seemann

CSS: Target abfragen / Modal implementieren

- In CSS können wir auf eine Sprungmarke in der URL reagieren
- Damit können wir unser Modal implementieren!
- Zusätzlich dazu werden wir uns anschauen, wie wir "eigene" Formularelemente per CSS designen können
- **Zudem (Bonus):**
  - Einige Browser (wie z.B. Chrome oder Safari) unterstützen einen schicken Hintergrund-Effekt
  - Wenn wir hierfür ein funktionierendes Fallback einbauen, können wir diesen schon jetzt nutzen!

Fortgeschrittenes CSS und SASS

Seiten-Menü

Jannis Seemann

Seiten-Menü

- In diesem Abschnitt geht es um das Seiten-Menü
- Dieses ist komplett in CSS implementiert
  - Inklusive Animation zum Ein- bzw. Ausblenden!
- Das ist eine weitere, super Anwendung für den Checkbox-Trick
- **Wir werden hier aber auf ein Problem stoßen:**
  - Damit die Animation funktioniert, dürfen wir das Seitenmenü nicht per display: none ausblenden
  - Diese Eigenschaft lässt sich nicht animieren
  - Wir müssen dafür also eine Lösung finden...

Fortgeschrittenes CSS und SASS

Fußbereich

Jannis Seemann

Fußbreich

- In diesem Abschnitt geht es um den Fußbereich
- Hierbei wenden wir primär existierendes Wissen an
- **Allerdings:**
  - Es ist ein super Beispiel, wie wir Flexboxen ineinander verschachteln können
- **Das bedeutet für dich:**
  - Du kannst diesen Abschnitt überspringen...
  - ... oder dich eigenständig am Fußbereich versuchen
  - ... oder dir die Implementierung von mir anschauen

Fortgeschrittenes CSS und SASS

Finaler Feinschliff

Jannis Seemann

Finaler Feinschliff

- Mein Anspruch ist: Genau das Design, was ich dir am Anfang des Kurses gezeigt habe, entwickeln wir vollständig hier in diesem Kurs
- Es fehlt aber noch ein bisschen was an Feinschliff
- Teilweise funktionieren Dinge auf Smartphone nicht (Video spielt nicht ab)...
- oder wir können das Responsive-Design noch etwas optimieren
- **Aber:**
  - Hier geht es primär um Feinschliff
  - Bis auf den Smartphone-Bug (separate Lektion) geht es nicht um "neue Dinge"
  - Betrachte diese Lektionen daher bitte als optional

Fortgeschrittenes CSS und SASS

Bonus: SVG-Sprites

Bonus: SVG-Sprites

- Bisher waren alle Icons, etc. direkte SVG-Elemente in unserer HTML-Datei
- Das ist super für eine One-Page-Webseite
- Aber wenn wir mehrere Unterseiten haben, kann eine SVG-Sprite Sinn machen
- Eine SVG-Sprite ist eine separate SVG-Datei, welche mehrere SVG-Bilder enthält
- Und aus dem HTML-Code wird dann ein einzelnes Icon eingebettet
- **Probleme:**
  - Diese SVG-Sprite funktioniert etwas anders als ein inline-SVG
  - Wir müssen dies beim Styling mit CSS beachten
  - Zusätzlicher Aufwand: Diese SVG-Sprite muss erstellt werden
  - Browser laden die Datei nur über http(s)://, nicht aber wenn wir die .html-Datei direkt öffnen

Fortgeschrittenes CSS und SASS

Bonus: Bilder komprimieren

Bonus: Bilder komprimieren

- Ein Responsive-Design passt sich der Bildschirmgröße an
- Ein Bild wird daher oft unterschiedlich groß angezeigt
- Bisher wird aber immer exakt das gleiche .jpg-Bild geladen
- **Lösung:**
  - srcset-Attribut: Browser entscheidet, welches Bild er laden möchte
  - <picture>-Tag: Damit können wir dem Browser auch eine .webp-Version vom Bild anbieten
- **Probleme:**
  - Die Schreibweise ist sehr viel aufwendiger
  - Wir müssen verschiedene Versionen von unseren Bildern generieren lassen

Fortgeschrittenes CSS und SASS

Schlussworte

Was hast du gelernt?

- Du behältst dank SASS auch über riesige Projekte den Überblick
- Du kannst die Ladezeit deiner Seite optimieren
- Du kannst komplexe Interaktionen komplett per CSS entwickeln
- Für einige Dinge, die du früher mit JavaScript entwickelt hättest, reichen jetzt ein paar Zeilen CSS-Code (z.B. Checkbox-Trick)
- Du kannst auf eingefärbte Hintergrundbilder, etc. verzichten, und den gleichen Effekt per CSS erreichen
- Du verwendest moderne CSS-Technologien, für noch schickere Designs:
  - CSS Flexbox
  - CSS Grid
  - Filter, transition, transform
  - Responsive Design (@media,...)

Wie geht es jetzt weiter?

- Du bist jetzt ein echter CSS und SASS-Profi
- Aber wie geht es jetzt weiter?
- Das hängt davon ab, was du erreichen möchtest
- Wenn noch nicht geschehen, jetzt könnte ein guter Zeitpunkt sein, dich weiter mit JavaScript oder PHP zu beschäftigen
- Und/oder du lernst den Umgang mit CMS-Systemen, wie z.B. Wordpress, Magento oder Shopware
- Das sind dann Dienstleistungen, die du verkaufen kannst

Vielen Dank!