

BOOTSTRAP 4

EINFÜHRUNG UND THEMING

EINFÜHRUNGSTHEMEN

- **Einstieg in Bootstrap 4**
Ressourcen und Download
Quellen für Themes
Browser-Kompatibilität
Layout mit CSS Klassen
Erweiterungen und Plugins
- **Umgang mit dem Grid-System**
Gestaltungsprinzipien und
12er-Raster
Zwischenräume
Automatisches Layout
Responsive Klassen
Ausrichtung und Sortierung
Verschachtelte Darstellungen
Flexbox horizontal und vertikal
Suchmaschinen optimierter
HTML-Code
- **Typografie**
Größen einstellen
Farbschemata anwenden
Abstände
Textausrichtung
Schatten und Rahmen
Sichtbarkeit und Textersetzung
- **Typische Inhalte verwenden**
Texte und Überschriften
Bilder und Medien-Elemente
Formulare und Tabellen
Icon Sets verwenden
- **Navigationen erstellen**
Responsive Navbars
Menüpunkte als Dropdown
Tabbed Navigation
- **Komponenten verwenden**
Alerts, Badges und Buttons
Listen
Cards und Jumbotron
- **Interaktive Komponenten**
Pagination
Progress
Scroll Spy
Modal Boxen, Popup-Fenster
Karussell, Slider
Akkordeon
Tool-Tipps
- **Bootstrap und JavaScript**
jQuery und popper.js
Data Attribute, data-toggle
Events und asynchrone
Funktionen
Defaults überschreiben
Methoden der Komponenten
- **Theming**
Überschreiben der CSS-
Klassen
Eigenes Bootstrap-CSS
erstellen mit Online-Tools

AUFBAUTHemen

- **Einführung**

Übersicht über die Projektstruktur
npm Befehle einsetzen
Einen eigenen Build erstellen
Dateigrößen optimieren
QUnit Test ausführen

- **SCSS Einführung**

Umwandlung von SCSS in CSS
Verwendung von Variablen
Verschachtelte Klassen
Stile auslagern und importieren
Mixins erstellen
Erweitern und erben von Eigenschaften
Operatoren anwenden
@Rules und Direktiven
Bedingte Stile
Maps und Schleifen

- **Bootstrap und SCSS**

Variablen für Farben und Grautöne
Variablen für Media Queries
Wichtige SCSS Funktionen
Bootstrap Optionen ändern

- **Anwendungsfälle**

Eigene Farben verwenden
Default Variablen überschreiben
Spaltenanzahl und Abstände im Grid ändern
Media Queries ändern
Eigene Komponenten erstellen



Richte deine Arbeitsumgebung ein.

Lies dazu das Readme auf <https://github.com/zenbox/workshop/> und folge den Anweisungen.

```
workshop/
├── node_modules/
│   └── bootstrap
│       ├── dist
│       ├── js
│       └── scss
├── assets
│   ├── css
│   └── scss
│       └── main.scss
├── index.html
├── package-lock.json
└── package.json
```

15 MIN



Richte deine Arbeitsumgebung ein.

Lies dazu das Readme auf <https://github.com/zenbox/workshop/> und folge den Anweisungen.

```
workshop/
├── node_modules/
│   └── bootstrap
│       ├── dist
│       ├── js
│       └── scss
├── assets
│   ├── css
│   └── scss
│       └── main.scss
├── index.html
├── package-lock.json
└── package.json
```

15 MIN

LEARNING SASS BASICS

IMPORTS

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

```
// main.scss
@import 'reset';
```

```
$ sass --watch assets/scss:assets/css
-> main.css
```

VARIABLES

```
// LAYOUT
$default-padding: 0.5rem;
$default-margin: 0;
```

```
// FONTS
$font-size: 22px;
$line-height: 1.8rem;
```

```
// COLORS
$light: #ffe;
$dark: #000;
```


MIXINGS

```
@mixin transform($property) {  
  -webkit-transform: $property;  
  -ms-transform: $property;  
  transform: $property;  
}
```


```
.box { @include transform(rotate(30deg)); }
```

MIXINGS II

```
@mixin corner-icon($name, $top-or-bottom, $left-or-right) {  
  .icon-#{#{ $name }} {  
    background-image: url("/icons/#{ $name }.svg");  
    position: absolute;  
    #{ $top-or-bottom }: 0;  
    #{ $left-or-right }: 0;  
  }  
}  
  
@include corner-icon("mail", top, left);  
  
.icon-mail { background-image: url("/icons/mail.svg"); }
```

NESTING

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



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

NESTING WITH COMBINATORS

```
ul > {  
  li {  
    list-style-type: none;  
  }  
}  
  
h2 {  
  + p {  
    border-top: 1px solid gray;  
  }  
}  
  
p {  
  ~ {  
    span {  
      opacity: 0.8;  
    }  
  }  
}
```

```
ul > li { ... }  
  
h2 + p { ... }  
  
p ~ span { ... }
```

NESTING WITH PARENT SELECTOR



```
.alert {  
  &:hover {  
    font-weight: bold;  
  }  
  
  [dir=rtl] & {  
    margin-left: 0;  
    margin-right: 10px;  
  }  
  
  :not(&) {  
    opacity: 0.8;  
  }  
}
```



```
.alert:hover { ... }  
.alert[dir=rtl] { ... }  
:not(.alert) { ... }
```

MODULES

```
// _base.scss
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}

// main.scss
@use 'base';

.inverse {
  background-color: base.$primary-color;
  color: white;
}
```

EXTEND

```
%message-shared {  
  border: 1px solid #ccc;  
  padding: 10px;  
  color: #333;  
}  
  
.message {  
  @extend %message-shared;  
}  
  
.success {  
  @extend %message-shared;  
  border-color: green;  
}
```

OPERATORS

```
.container {  
  width: 100%;  
}  
  
article[role="main"] {  
  float: left;  
  width: 600px / 960px * 100%;  
}  
  
aside[role="complementary"] {  
  float: right;  
  width: 300px / 960px * 100%;  
}
```

CSS4: `calc(600px/960px*100%)`

FUNCTIONS

```
@function pow($base, $exponent) {  
  $result: 1;  
  @for $_ from 1 through $exponent {  
    $result: $result * $base;  
  }  
  @return $result;  
}  
  
.sidebar {  
  float: left;  
  margin-left: pow(4, 3) * 1px;  
}
```

IF ELSE

```
$light-background: #f2ece4;
$light-text: #036;
$dark-background: #6b717f;
$dark-text: #d2e1dd;

@mixin theme-colors($light-theme: true) {
  @if $light-theme {
    background-color: $light-background;
    color: $light-text;
  } @else {
    background-color: $dark-background;
    color: $dark-text;
  }
}

.banner {
  @include theme-colors($light-theme: true);
  body.dark & {
    @include theme-colors($light-theme: false);
  }
}
```

EACH

```
$sizes: 40px, 50px, 80px;
```

```
@each $size in $sizes {
```

```
  .icon-#{ $size } {  
    font-size: $size;  
    height: $size;  
    width: $size;
```

```
  }
```

```
}
```

EACH WITH MAPS

```
$icons: ("eye": "\f112", "start": "\f12e", "stop": "\f12f");

@each $name, $glyph in $icons {
  .icon-#{$name}:before {
    display: inline-block;
    font-family: "Icon Font";
    content: $glyph;
  }
}
```

FOR

```
$base-color: #036;
```

```
@for $i from 1 through 3 {
```

```
  ul:nth-child(3n + #{ $i }) {
```

```
    background-color: lighten($base-color, $i * 5%);
```

```
  }
```

```
}
```

WHILE

```
// Divides `$value` by `$ratio` until it's below `$base`.  
@function scale-below($value, $base, $ratio: 1.618) {  
  @while $value > $base {  
    $value: $value / $ratio;  
  }  
  
  @return $value;  
}  
  
$normal-font-size: 16px;  
sup {  
  font-size: scale-below(20px, 16px);  
}
```



Arbeitsgruppen:

Erprobt verschiedene (mindestens 5) SASS Funktionalitäten anhand von selbst erdachten Anwendungsfällen und stellt sie anschließend kurz vor.

Zum Beispiel: **variables**, **mixings**, **nesting with parent**, **for** und **extend**.

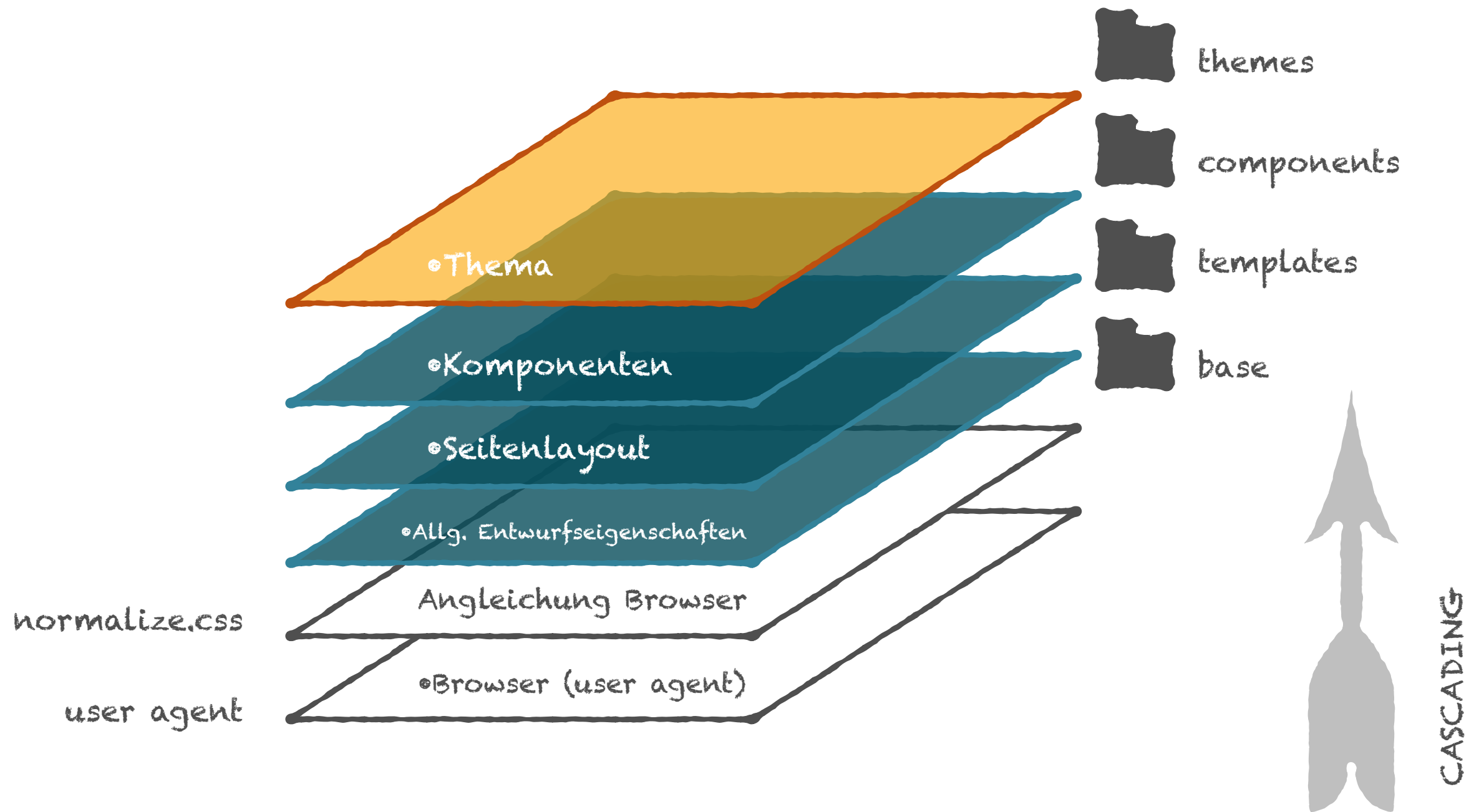
```
workshop/  
├── assets  
│   ├── css  
│   │   └── sass-examples.css  
│   └── scss  
│       └── sass-examples.scss  
└── index.html
```

[HTTPS://SASS-LANG.COM/DOCUMENTATION/](https://sass-lang.com/documentation/)

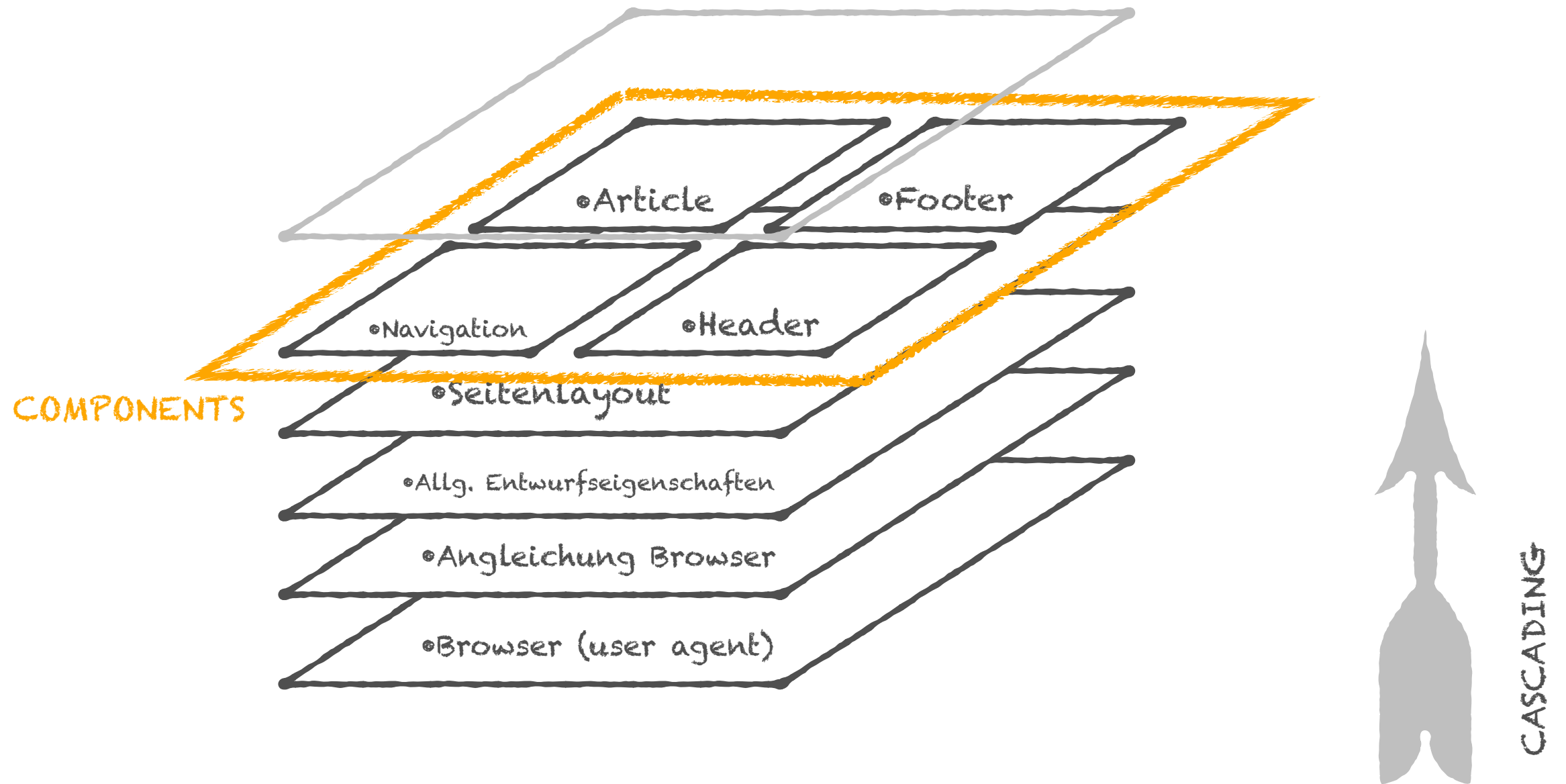
20 MIN

CASCADING STYLESHEETS ARCHITECTURE

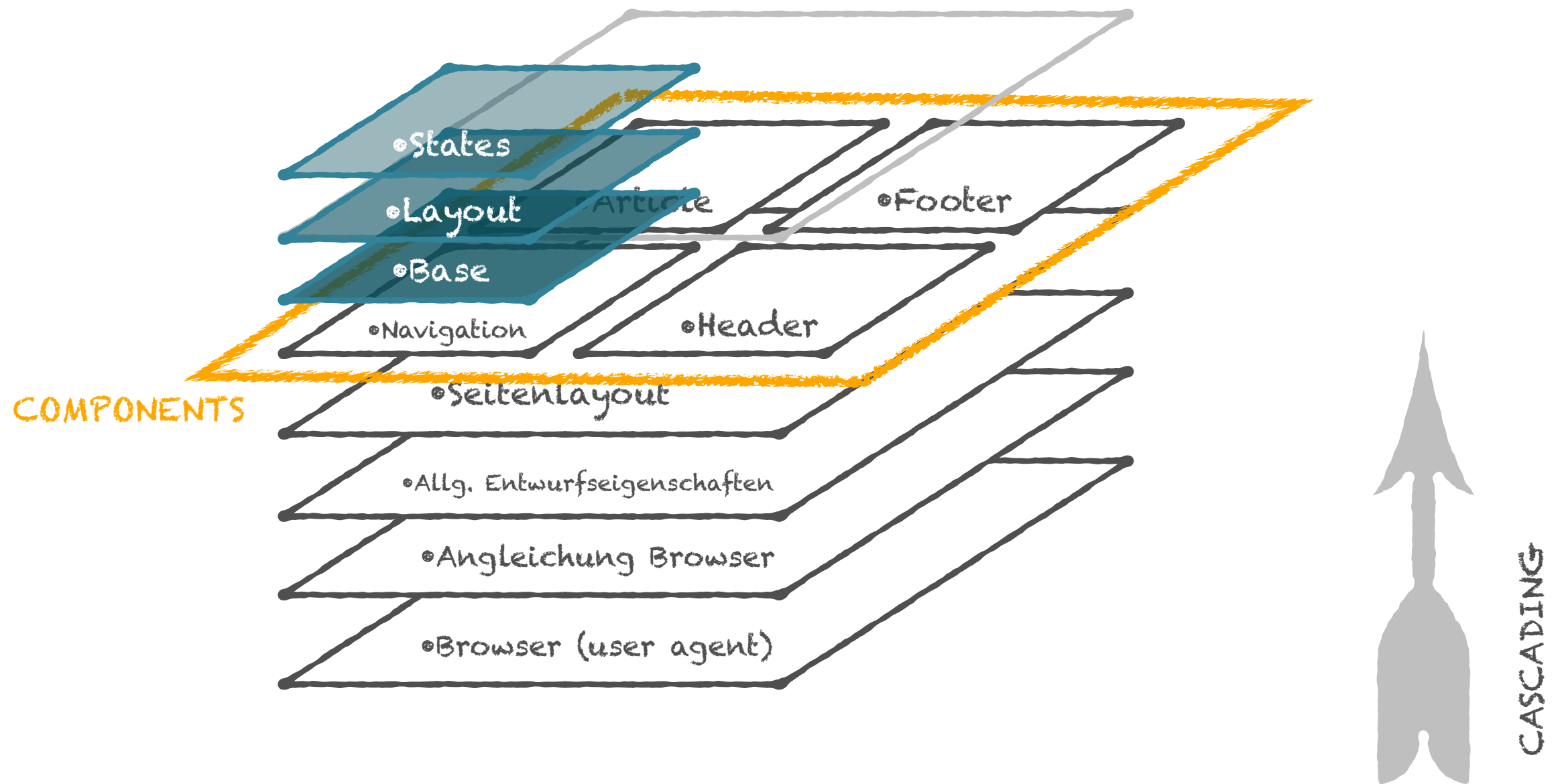
CSS ARCHITECTURE



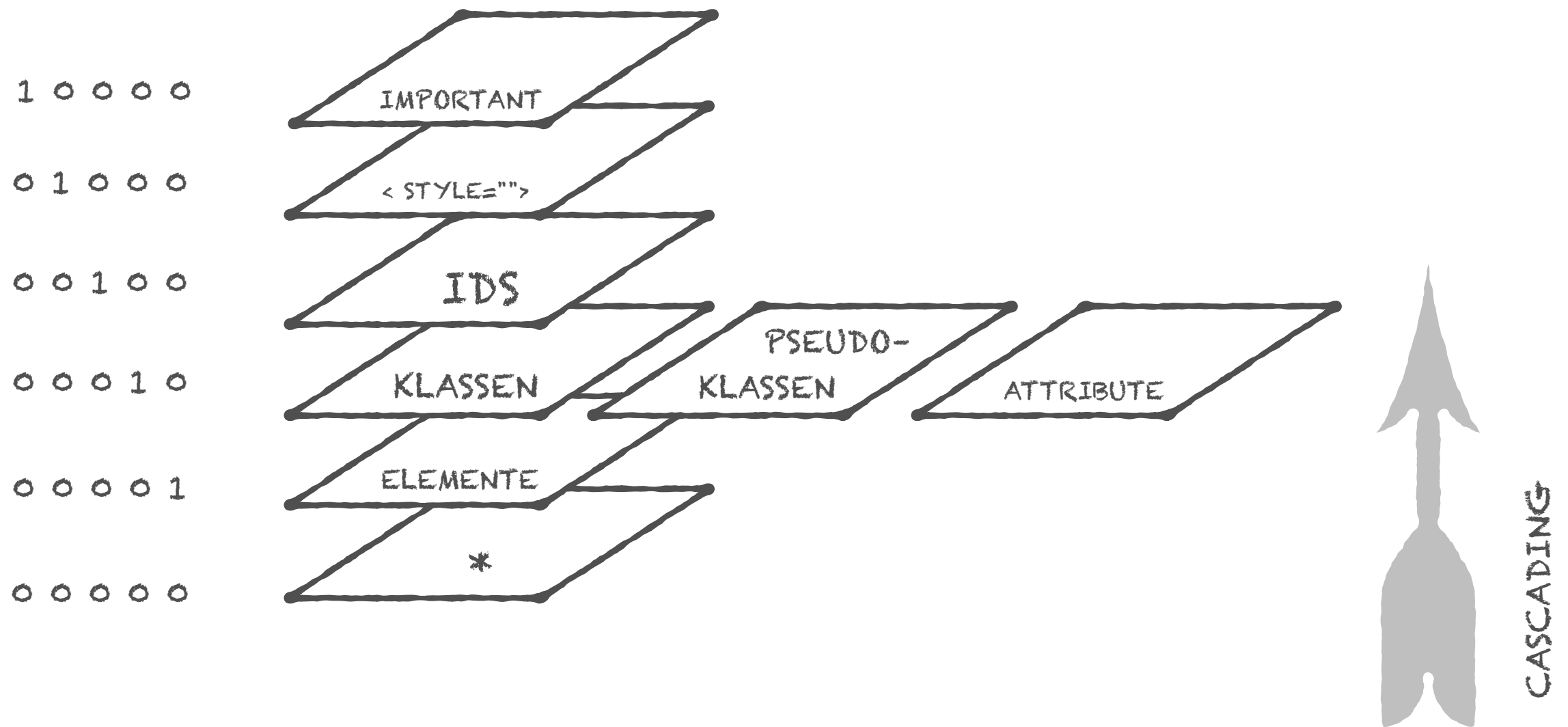
CSS CASCADE



CSS CASCADE



CSS GEWICHTUNG



🕒 IMPORTANT
ODER < STYLE="" >

🕒 PSEUDOKLASSEN

KLASSEN

🕒 PSEUDOKLASSEN

ELEMENTE

0
0
1
0
0
0
0
0

0
0
0
0
0
0
0
1

gen. Zustände

•Thema

States
Layout
Base

Komponenten

Seitenlayout

Allg. Zustände

Allg. Entwurf

Angleichung Browser

Browser (user agent)



CASCADING



Wende die CSS Architektur an.

```
workshop/
├── assets
│   ├── css
│   │   └── main.css
│   └── scss
│       ├── _variables.scss
│       ├── _mixins.scss
│       ├── _my-theme-base.scss
│       ├── _my-theme-template.scss
│       ├── _my-theme-component-1.scss
│       └── main.scss
└── index.html
```

15 MIN

THEMING BOOTSTRAP

IMPORT SEQUENCE

```
// 1. Own variables to override bootstrap variables
@import "variables";

// 2. Bootstrap or bootstrap files
@import "../..../node_modules/bootstrap/scss/bootstrap.scss";

// 3. Own mixins
@import "mixins";

// 4. Own theme files
@import "base";
@import "template";

@import "components/icons";
@import "components/nav";
@import "components/sample";
@import "components/tables";
```


[https://getbootstrap.com/docs/4.4/
getting-started/theming/](https://getbootstrap.com/docs/4.4/getting-started/theming/)

[GETBOOTSTRAP.COM](https://getbootstrap.com)

1. ADDITIONAL CLASSES

Needs the same specificity to override BS classes.
To extend BS classes, the specificity doesn't matter.

```
.table .thead-dark th {  
  color: white;  
}  
  
.my-own-class {  
  color: black;           /* Will not work! */  
  background-color : white; /* Works since the BS class has  
                           no background-color property */  
}
```

2. REWRITE OR EXTEND BOOTSTRAP COMPONENTS

You need to know about the bs classes to get the same specificity, i.e. by using the same SASS nesting.

```
.table {  
  .thead-dark {  
    ...  
    th { ... }  
  }  
}
```

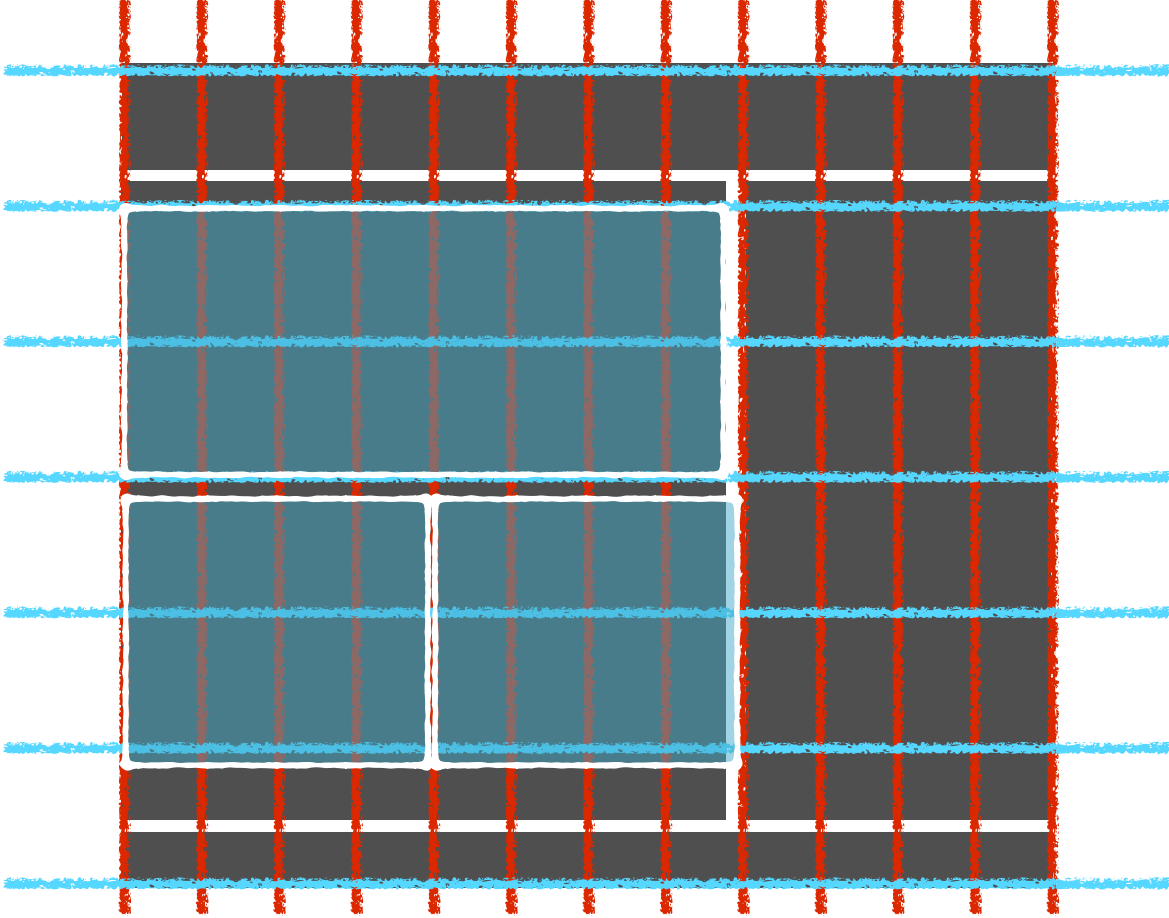
```
.my-table {  
  ...  
  .my-thead {  
    th { ... }  
  }  
}
```

3. SET NEW VALUES TO THE BOOTSTRAP VARIABLES

```
$font-family-sans-serif: Roboto, Arial, sans-serif;  
$font-family-monospace: Menlo, "Courier New", monospace;
```

VARIABLES SECTIONS

- **Color system**
- **Spacing**
Control the default styling of most Bootstrap elements by modifying these variables. Mostly focused on spacing. You can add more entries to the `$spacers` map, should you need more variation.
- **Body**
Settings for the `<body>` element.
- **Links**
Style anchor elements.
- **Paragraphs**
Style `p` element.
- **Grid breakpoints**
Define the minimum dimensions at which your layout will change, adapting to different screen sizes, for use in media queries.
- **Grid containers**
Define the maximum width of `.container` for different screen sizes.
- **Grid columns**
Set the number of columns and specify the width of the gutters.
- **Components**
Define common padding and border radius sizes and more.
- **Typography**
Font, line-height, and color for body text, headings, and more.
- **Tables**
Customizes the `.table` component with basic values, each used across all table variations.
- **Buttons + Forms**
Shared variables that are reassigned to `$input-` and `$btn-` specific variables.
- **Buttons**
For each of Bootstrap's buttons, define text, background, and border color.
- **Forms and Form validation**
- **Component variables:** Navbar, Dropdowns, Pagination, Jumbotron, Cards, Tooltips, Popovers, Toasts, Badges, Modals, Alerts, Progress bars, List group, Image thumbnails, Figures, Breadcrumbs, Carousel, Spinners, Close, Code



MODIFY THE DEFAULT BREAKPOINTS

```
$grid-breakpoints: map-merge(  
  (  
    xs: 0,  
    sm: 576px,  
    md: 768px,  
    lg: 992px,  
    xl: 1200px  
  ),  
  $grid-breakpoints  
);
```

To update the existing breakpoints or add new ones, you just need to override the `$grid-breakpoints` variable passing only the required keys.

```
$grid-breakpoints: (  
  sm: 500px,  
  xxl: 1440px  
);
```

UPDATE THEME COLORS

```
$blue: #007bff !default;
$indigo: #6610f2 !default;
$purple: #6f42c1 !default;
$pink: #e83e8c !default;
$red: #dc3545 !default;
$orange: #fd7e14 !default;
$yellow: #ffc107 !default;
$green: #28a745 !default;
$teal: #20c997 !default;
$cyan: #17a2b8 !default;

[...]

$primary: #20c997; // your custom primary color

@import "bootstrap/functions";
@import "bootstrap/variables";
@import "bootstrap/mixins";
[...]
W
```


CUSTOMIZE THE SPACING RULES

```
$spacer: 1rem;
```

```
$spacers: (  
  1: ($spacer * .5),  
  2: $spacer,  
  3: ($spacer * 1.5),  
  4: ($spacer * 2),  
  5: ($spacer * 3),  
  6: ($spacer * 5)  
);
```

Will produce the following CSS:

```
.m-0 { margin: 0 !important;  
}  
[...]  
.m-1 { margin: 0.5rem !important; }  
[...]  
.m-2 { margin: 1rem !important; }  
[...]  
.m-3 { margin: 1.5rem !important; }  
[...]  
.m-4 { margin: 2rem !important; }  
[...]  
.m-5 { margin: 3rem !important; }  
[...]  
  
// new one  
.m-6 { margin: 5rem !important; }  
[...]
```

SOME USEFUL VARIABLES

`$enable-caret: true;`

is used to manage the caret icon displayed mainly within the dropdown elements.
Default value is true.

`$enable-rounded: true;`

is used to control the rounded borders of elements.
If displaying rounded borders you can also manage the border-radius value
overriding the `$border-radius` variable.

`$enable-shadows: false;`

is used to control the box-shadow property of buttons and form elements.
`$btn-box-shadow` is the variable that controls the box-shadow styling.

`$enable-gradients: false;`

enables predefined gradients via background-image styles on various components.

`$enable-transitions: true;`

`$enable-prefers-reduced-motion-media-query: true;`

`$enable-grid-classes: true;`

`$enable-print-styles: true;`

`$enable-validation-icons: true;`

[https://getbootstrap.com/docs/4.2/getting-started/
theming/#sass-options](https://getbootstrap.com/docs/4.2/getting-started/theming/#sass-options)

GETBOOTSTRAP.COM

<https://uxplanet.org/how-to-customize-bootstrap-b8078a011203>

MEDIUM.COM



Überschreibe einige Grundfarben, erfinde eine neue Typografie, schreibe ein neues Navbar Theme, setze die Spacer neu.

```
workshop/
├── assets
│   ├── css
│   │   └── main.css
│   └── scss
│       ├── _base.scss
│       ├── _colors.scss
│       ├── _typography.scss
│       ├── _variables.scss
│       └── main.scss
└── index.html
```

Eigene Komponenten mit eigenen Funktionen schreiben.
Grid Verständnis



Schreibe ein neues Tabellen Theme.

```
workshop/
├── assets
│   ├── css
│   │   └── main.css
│   └── scss
│       ├── components
│       │   ├── _tables.scss
│       │   └── _navbar.scss
│       ├── _base.scss
│       ├── _variables.scss
│       └── main.scss
└── index.html
```

45 MIN