Shopware 6 Theme entwickeln – SCSS and Styling

Shopware 6 Theme entwickeln – SCSS and Styling

Shopware 6: SCSS and Styling

In our documentation you will find all the information you need for your daily work with Shopware.

[expand title="mehr lesen…"]

The stylesheets are written in SASS. The folder structure is inspired by the 7-1 pattern structure.

```
<platform/src/Storefront/Resources/app/storefront/src/scss>
└── abstract
└── base
└── component
└── layout
└── page
└── skin
└── vendor
└── base.scss
```

Shopware 6 looks inside your theme.json file to find a „style“ array which contains all SCSS files which should be loaded by your theme. By default you get the Shopware Storefront SCSS plus an additional entry point for your own SCSS. You can also extend this array with more SCSS files.

```
{
  "name": "Just another theme",
```

```
  ...

  "style": [
    "@Storefront",
    "app/storefront/src/scss/base.scss" <-- Theme SCSS entry
  ],

  ...

}
```

To try it out, open up the base.scss file from your theme.

Inside of the .scss file, add some styles like this:

```
// src/Resources/app/storefront/src/scss/base.scss
body {
    background: blue;
}
```

To see if it’s working you have to re-build the storefront. Use the following command to do that.

```
# run this to re-compile the current storefront theme
$ ./psh.phar storefront:build

# or run this to start a watch-mode (the storefront will re-compile when you make sytle changes)
$ ./psh.phar storefront:hot-proxy
```

In this example, the background of the body will be changed.

Working with variables

In case you want to use the same color in several places, but want to define it just one time you can use variables for this.

Create a abstract/variables.scss file inside your „scss“ folder and define your background color variable.

```
// in variables.scss
```

```
$sw-storefront-assets-color-background: blue;
```

Inside your base.scss file you can now import your previously defined variables and use them:

```
// in base.scss
@import 'abstract/variables.scss';

body {
    background: $sw-storefront-assets-color-background;
}
```

This has the advantage that when you want to change the values of your variables you just have one location to change them and the hard coded values are not cluttered all over the codebase.

Bootstrap

The storefront theme is implemented as a skin on top of the Boostrap toolkit.

Override default SCSS variables

To override default variables like for example $border-radius from Bootstrap you should use a slightly different approach then explained in the storefront assets how-to.

Bootstrap 4 is using the !default flag for it’s own default variables. Variable overrides have to be declared beforehand.

More information: https://getbootstrap.com/docs/4.0/getting-started/theming/#variable-defaults

To be able to override Bootstrap variables you can define an additional SCSS entry point in your theme.json which is declared before @Storefront. This entry point is called overrides.scss:

```
{
  "name": "Just another theme",
  "author": "Just another author",
  "views": [
     "@Storefront",
     "@Plugins",
     "@JustAnotherTheme"
  ],
  "style": [
     "app/storefront/src/scss/overrides.scss", <-- Variable
overrides
    "@Storefront",
    "app/storefront/src/scss/base.scss"
  ],
  "script": [
    "@Storefront",
    "app/storefront/dist/storefront/js/just-another-theme.js"
  ],
  "asset": [
    "app/storefront/src/assets"
  ]
}
```

In the overrides.scss you can now override default variables like $border-radius globally:

```
/*
Override variable defaults
================================================
This file is used to override default SCSS variables from the
Shopware Storefront or Bootstrap.

Because of the !default flags, theme variable overrides have
to be declared beforehand.
https://getbootstrap.com/docs/4.0/getting-started/theming/#var
iable-defaults
*/

$disabled-btn-bg: #f00;
$disabled-btn-border-color: #fc8;
$font-weight-semibold: 300;
$border-radius: 0;
```

```
$icon-base-color: #f00;
$modal-backdrop-bg: rgba(255, 0, 0, 0.5);
```

Please only add variable overrides in this file. You should not write CSS code like .container { background: #f00 } in this file. When running storefront:hot or storefront:hot-proxy SCSS variables will be injected dynamically by webpack. When writing selectors and properties in the overrides.scss the code can appear multiple times in your built CSS.

Using Bootstrap SCSS only

The Shopware default theme is using Bootstrap with additional custom styling.

If you want to build your theme only upon the Bootstrap SCSS you can use the @StorefrontBootstrap placeholder instead of the @Storefront bundle in the style section of your theme.json. This gives you the ability to use the Bootstrap SCSS without the Shopware Storefront „skin“. Therefore all the SCSS from src/Storefront/Resources/app/storefront/src/scss/skin wil l not be available in your theme.

```
{
  "style": [
    "@StorefrontBootstrap",
    "app/storefront/src/scss/base.scss"
  ]
}
```

- This option can only be used in the style section of the theme.json. You must not use it in views or script.
- All theme variables like $sw-color-brand-primary are also available when using the Bootstrap option.
- You can only use either @StorefrontBootstrap or @Storefront. They should not be used at the same time.

The @Storefront bundle **includes** the Bootstrap SCSS already.

[/expand]