# Shopware 6 Theme entwickeln – Twig templates

# Shopware 6 Theme entwickeln – Twig templates

[expand title="mehr lesen…"]

In Shopware 6 we use the Twig template engine to render the HTML in the storefront.

The templates can be found in platform/src/Storefront/Resources/views/storefront/ Here is a brief overview of the structure and the most important template sections:

```
#
vendor/shopware/platform/src/Storefront/Resources/views/storefront
├── base.html.twig
│
├── block       # Part of the content management system
│   ├── cms-block-category-navigation.html.twig
│   ├── cms-block-center-text.html.twig
│   ├── cms-block-form.html.twig
│   ├── ...
│
├── element     # Part of the content management system
│   ├── cms-element-category-navigation.html.twig
│   ├── cms-element-form
│   ├── cms-element-form.html.twig
│   ├── ...
│
├── section     # Part of the content management system
│   ├── cms-section-block-container.html.twig
```

```
│   ├── cms-section-default.html.twig
│   └── cms-section-sidebar.html.twig
│
├── component   # Shared content templates form the basis of the pages.
│   ├── account
│   ├── address
│   ├── analytics.html.twig
│   ├── checkout
│   ├── ...
│
├── layout       # Layout templates. Navigation, header and footer content templates are located here.
│   ├── breadcrumb.html.twig
│   ├── cookie
│   ├── footer
│   ├── header
│   ├── ...
│
├── page        # The concrete templates rendered by the page controllers. This directory contains full page templates as well as private local includes and the pagelet ajax response templates if necessary.
│   ├── account
│   ├── checkout
│   ├── content
│   ├── error
│   ├── newsletter
│   ├── product-detail
│   ├── search
│   └── sitemap
│
└── utilities   # Technical necessities used across the content, header and footer sections across all domain concepts.
    ├── alert.html.twig
    ├── form-violation.html.twig
    ├── icon.html.twig
    ├── ...
```

The base.html.twig is the root file of the storefront which holds every rendered component.

So, for instance, if you would like to modify the header, you would want to recreate the specific directory structure in order to be able to overwrite or extend the already existing elements. The storefront header in the header.html.twig file (which is later included into the base.html.twig) is located inside
the platform/src/Storefront/Resources/views/storefront/layout/header directory.

# Blocks

Almost every HTML chunk is wrapped into a corresponding Twig-Block, so it is easy to overwrite or extend existing blocks.

# Adding new content by extending the template

In this example we want add some extra content like the shop name below the logo. To do so, you need to extend the existing logo.html.twig file which is located inside the platform/src/Storefront/Resources/views/storefront/layout/header directory.

First we create a new file inside our theme.

```
# move into your theme folder
$ cd custom/plugins/MyTheme

# create coresponding folder structure
$ mkdir -p src/Resources/app/storefront/views/layout/header

# create a new file
$ touch src/Resources/app/storefront/views/layout/header/logo.html.twig
```

Then we extend the existing file through the sw_extends command.

```
# src/Resources/app/storefront/views/layout/header/logo.html.twig

{% sw_extends '@Storefront/storefront/layout/header/logo.html.twig' %}
```

After that you save this new file. Everything should be rendered like before.

To add new content into to template you can override the layout_header_logo_link block from the logo.html.twig like this.

```
# src/Resources/app/storefront/views/layout/header/logo.html.twig

{# extend the original twig file #}
{% sw_extends '@Storefront/storefront/layout/header/logo.html.twig' %}

{# override the original twig block #}
{% block layout_header_logo_link %}

    {#
        call the `parent` function to keep the old behavior,
        otherwise the block gets overridden
    #}
    {{ parent() }}

    {# modified content added to the block #}
    <span>Company name of shop owner</span>

{% endblock %}
```

# Reuse of existing elements

If you build your own feature and you need e.g. an element to display the price of the current product you can include existing partials with sw_include like this.

```
<div class="my-theme an-alternative-product-view">
        ...

        {% block component_product_box_price %}
            {# use sw_include to include template partials #}
                                                {%   sw_include
'@Storefront/storefront/component/product/card/price-
unit.html.twig' %}
        {% endblock %}

        ...
    </div>
```

# Template multi inheritance

Due to the plugin and theme system in Shopware 6 it is possible that one storefront template gets extended by multiple plugins or themes, but Twig does not allow multi inheritance out of the box. Therefore we created our own twig functions sw_extends and sw_include, that work like twigs native extends or include, except that they allow multi inheritance. So it is really important to use the sw_extends and sw_include, instead of the native extends and include.

[/expand]