








# Template Editor

The purpose of the editor is to provide an easy way to create extensive and complete (master) template pages, without the use of the builder. All template pages are written in yaml and currently contain as fixtures to be used by the builder-generator.

## Fixtures

The fixtures can be found here or will be provided to you with a zip-file. When you open this folder, you will find the following yaml files, that consists of the pages and the theme file.

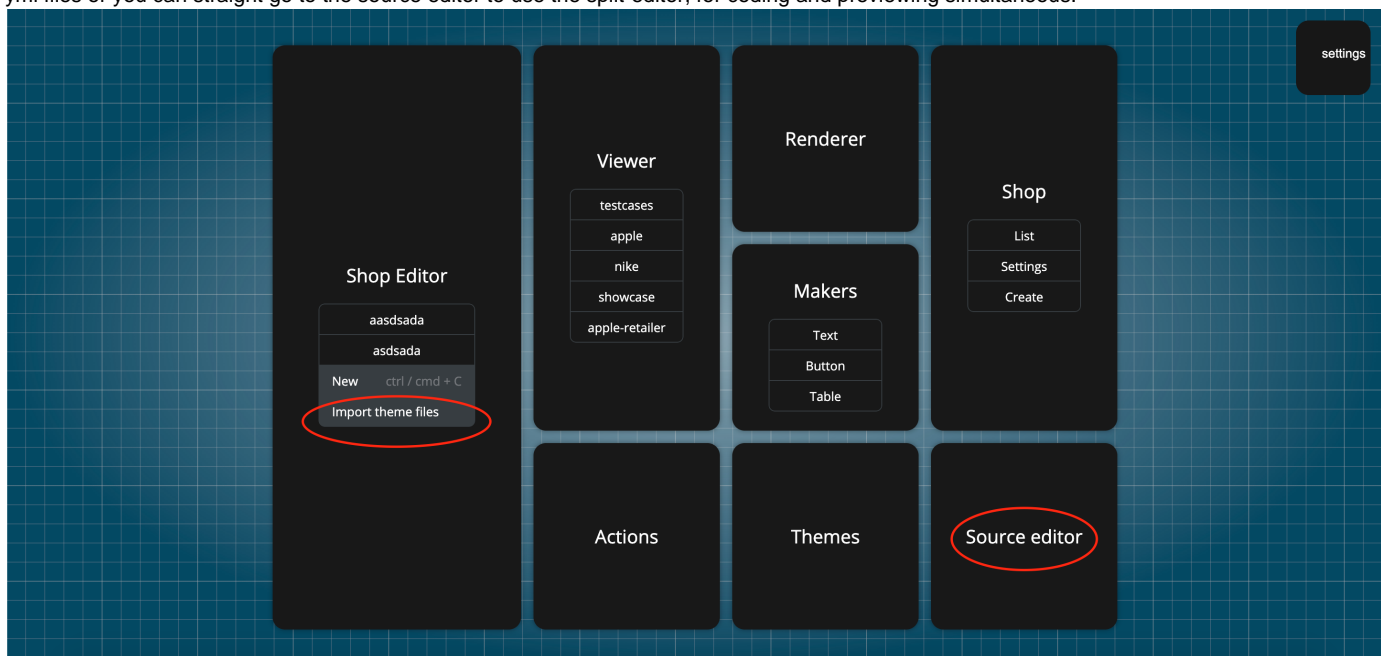
Name	Last commit	Last update
..		
 page.about.yaml	BEF-429: Fix nike	3 days ago
 page.category.yaml	BEF-429: Add nike about page	3 days ago
 page.contacts.yaml	BEF-468: Fix nike theme	3 days ago
 page.front.yaml	BEF-429: Add nike about page	3 days ago
 page.not-found.yaml	BEF-429: Add nike about page	3 days ago
 page.product.yaml	BEF-429: Add nike about page	3 days ago
 theme.yaml	BEF-468: Fix button, update themes	2 days ago

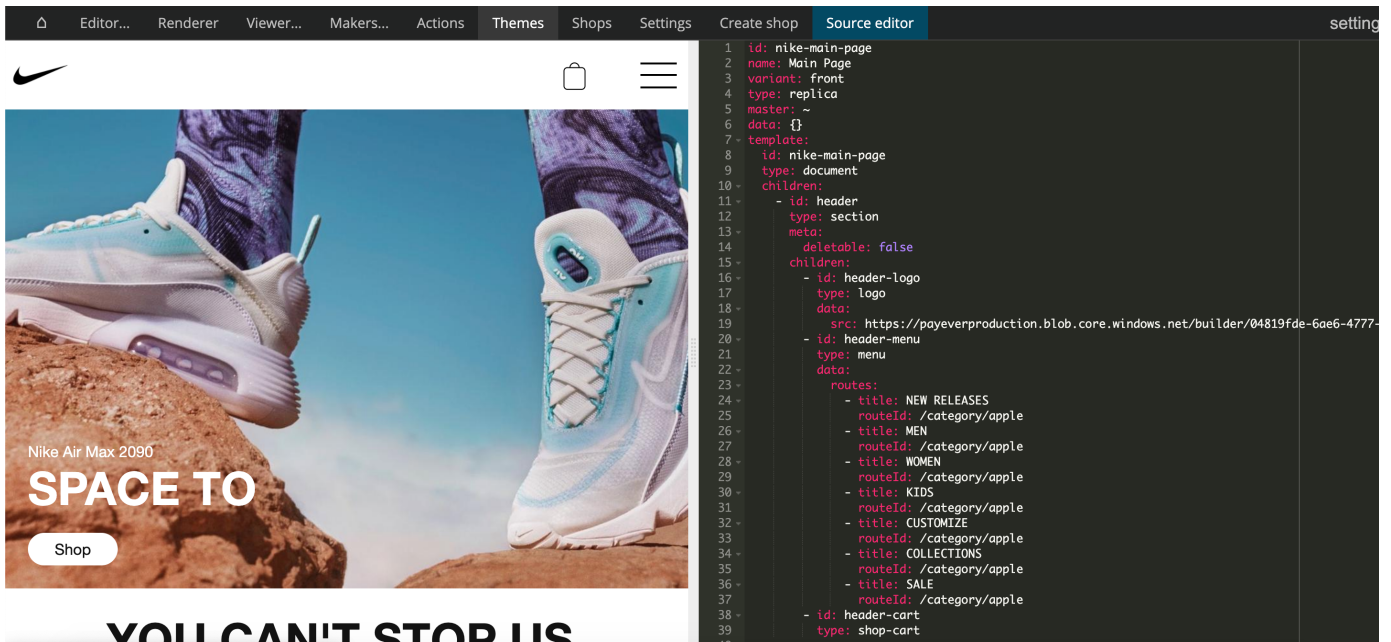
The theme.yaml file contains the aggregate information/routes and page names of the pages defined. The page . <page-name> .yaml represents the individual page template file.

## Editor

The source editor: <https://builder-editor-frontend.staging.devpayever.com/>

After you have opened the url, you will be presented with the editor. You can either use import theme files, to only 'view' the rendered form of the yaml files or you can straight go to the source editor to use the split-editor, for coding and previewing simultaneous.





## YML

### How can I import the yml file to the source editor?

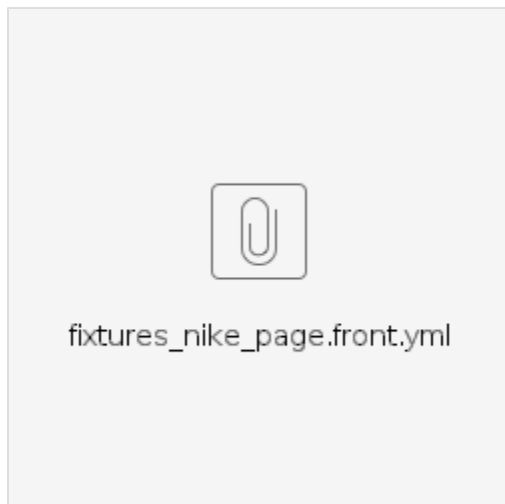
The easiest way would be opening the `yml` file in your local IDE, copy the plain text and paste it within the source editor on the url provided above. Once you have it in the editor, you will be able to adjust it and preview your changes on the left.

Screen Recording 2020-07-05 at 13.59.59.mov

### How to save the my changes?

After you have finalized your changes in the source editor, copy the entire plain text to your local IDE and save it as `page . <pagename> . yml`.

*Example test file*



## Sections

To provide proper distinction between each of the parts of a template page, it is important to implement sections. A page consists of multiple sections, e.g.:

- Header
- Hero

- CTA Section
- Body Section 1
- Body Section 2
- Pre-Footer
- Footer

Additionally it gives a more clean way to do styling, organising the content and simplicity of copying an entire section for reusing.

## Types

A template page defines a list of different `types`. A type defines what element you want to be represented in the page. See the list of all possible types below:

```
export const AVAILABLE_ELEMENTS_MAP = {
  document: PebDocumentElement,
  section: PebSectionElement,
  block: PebBlockElement,
  text: PebTextElement,
  grid: PebGridElement,
  line: PebLineElement,
  shape: PebShapeElement,
  button: PebButtonElement,
  html: PebHtmlElement,
  script: PebScriptElement,
  image: PebImageElement,
  menu: PebMenuElement,
  video: PebVideoElement,
  carousel: PebCarouselElement,
  'shop-products': PebShopProductsElement,
  'shop-product-details': PebShopProductDetailsElement,
  'shop-cart': PebShopCartElement,
  logo: PebLogoElement,
  'shop-category': PebShopCategoryElement,
  'mobile-menu': PebMobileMenuElement,
};
```

An example of how to use a type:

```
- id: first-section
  type: section
  children:
    - id: first-section-image
      type: block
      children:
        - id: first-section-image-title
          type: text
          data:
            text: Nike Air Max 2090
        - id: first-section-image-description
          type: text
          data:
            text: SPACE TO
        - id: first-section-image-button
          type: button
          data:
            link:
              type: internal
              routeId: /category/apple
            text: Shop
```

## Styles

The styles that are defined within the page just uses common CSS syntaxes, with the difference of using `semicolon` instead of `bracket`.

```

stylesheets:
  desktop:
    nike-main-page: {}
    # -----
    # Desktop Header
    # -----
    header:
      backgroundColor: '#fff'
      height: 44
      position: sticky
      top: 0
      zIndex: 1
      display: grid
      gridTemplateColumns: 100 1fr 100
    header-logo:
      width: 72
      height: 72
    header-menu:
      fontSize: 16
      fontWeight: 500
      fontFamily: Helvetica Neue,Helvetica,Arial,sans-serif
      color: '#000'
      margin: auto
    header-cart:
      width: 30
      height: 72
      margin: 0 0 0 55
      backgroundColor: '#fff'
      borderWidth: 3
      borderColor: '#000'

```

Each of the defined styles always starts with the id of the parent, e.g. - id: first-section, will use first-section: {}.

## Grid

Another important implementation necessity is the use of grid styles. Each of the sections/block requires to have gridTemplateRows/child-slots defined, without this our grid system and different device screens won't work.

```
first-section-image:
  display: grid
  gridTemplateRows: 3fr
  backgroundImage: 'https://payeverproduction.blob.core.windows.net
/builder/30620168-61f3-4961-8252-224ba3cb6633-nike-dream.jpg'
  backgroundSize: cover
  backgroundPosition: center
  height: 700
  width: 100%
```

### Desktop, Tablet, Mobile

In each individual page, within the stylesheets the styles have to be defined. We define styles for desktop, tablet and mobile. See the example below on how to distinct between the screens.

```
stylesheets:
  desktop:
    <desktop styles>
  tablet:
    <tablet styles>
  mobile:
    <mobile styles>
```

### Theme File

In order to link all created pages .<pagename>.yaml files into one single theme, provide its theme name and all routing that belong to it, these data have to be defined within the theme.yaml file.

Name, data, routing and pages are mandatory. Context of the logo is optional. Use below example as reference for making your own theme.yaml file.

```
name: Nike
data:
  productPages: /products/:productId
  categoryPages: /category/:categoryId
routing:
  - routeId: 'nike-main-page-route'
    url: /
    pageId: nike-main-page
  - routeId: 'nike-about-page-route'
    url: /about
    pageId: nike-about-page
  - routeId: 'nike-contacts-page-route'
    url: /contacts
    pageId: nike-contacts-page
  - routeId: 'nike-not-found-page-route'
    url: /404
    pageId: nike-not-found-page
  - routeId: 'nike-product-page-route'
    url: /products/:productId
    pageId: nike-product-page
  - routeId: 'nike-category-page-route'
    url: /category/:categoryId
    pageId: nike-category-page
context:
  '@logo':
    service: company
    method: getLogo
    params: []
pages:
  - front
  - product
  - category
  - about
  - contacts
  - not-found
```