

Theming in Drupal 8



David Hernandez

www.drupal.org/u/davidhernandez

@davidnarrabilis

Getting Started

- HTML5 by default
- Support for IE6/7/8 dropped
- Support for JQuery 2.0
- Twig is the new templating system
 - Comes from Symfony (<http://twig.sensiolabs.org/>)
- *.tpl.php => *.html.twig
- mytheme.info => mytheme.info.yml (<https://www.drupal.org/node/2349827>)
- template.php => mytheme.theme
- <https://www.drupal.org/theme-guide/8>
- <https://www.drupal.org/project/pistachio>

CSS and inclusion of SMACSS categorization

D8 uses the SMACSS method to organize CSS and files (<https://smacss.com/book/categorizing>)

- Base
- Layout
- Module => Changed to “Component” in Drupal for obvious reasons
- State
- Theme

CSS and inclusion of SMACSS categorization

```
/core/themes/bartik/css/base/elements.css?nyub3t" );  
/core/themes/bartik/css/layout.css?nyub3t" );  
/core/themes/classy/css/components/action-links.css?nyub3t" );  
/core/themes/classy/css/components/breadcrumb.css?nyub3t" );  
/core/themes/classy/css/components/button.css?nyub3t" );  
/core/themes/classy/css/components/collapse-processed.css?nyub3t" );  
/core/themes/classy/css/components/container-inline.css?nyub3t" );  
/core/themes/classy/css/components/details.css?nyub3t" );  
/core/themes/classy/css/components/exposed-filters.css?nyub3t" );  
/core/themes/classy/css/components/field.css?nyub3t" );  
/core/themes/classy/css/components/form.css?nyub3t" );  
/core/themes/classy/css/components/icons.css?nyub3t" );  
/core/themes/classy/css/components/inline-form.css?nyub3t" );  
/core/themes/classy/css/components/item-list.css?nyub3t" );
```

Libraries

- All JavaScript and CSS assets (files) are added using libraries
- Defined in mytheme.libraries.yml (<https://www.drupal.org/theme-guide/8/assets>)
- Unique name per module/theme
- Optional dependencies
- Optional version number

Libraries

```
# In mytheme.libraries.yml
```

```
global-styling:
```

```
  version: '1.0.x'
```

```
  css:
```

```
    # The SMACSS category.
```

```
    base:
```

```
      # The path to the css file.
```

```
      assets/css/base.css: {}
```

```
    theme:
```

```
      assets/css/print.css: { media: print }
```

```
  js:
```

```
    assets/js/myglobaljs.js
```

```
dependencies:
```

```
  - core/jquery
```

Libraries

```
# In mytheme.libraries.yml

# Add an external stylesheet.
#
# In the following example, we add a Google font (Lato).
lato-google-font:
  css:
    base:
      '//fonts.googleapis.com/css?family=Lato:300,400,700': { external: true }
```

Libraries

Other libraries can be manipulated using `libraries-extend` and `libraries-override`

```
# In mytheme.info.yml
```

```
libraries-extend:
```

```
# Classy's forums library is only included when the forums.html.twig  
# template is used. This will add my theme's forums library at the same  
# time.
```

```
classy/forums:
```

```
  - mytheme/forums
```


Libraries

```
# In mytheme.info.yml
```

```
libraries-override:
```

```
  # The library name.
```

```
  core/jquery.ui:
```

```
    # CSS files are always labeled as such. This format is required.
```

```
    css:
```

```
      # The SMACSS category is required.
```

```
      theme:
```

```
        # The path to the file. It is not a path relative to your theme.
```

```
        assets/vendor/jquery.ui/themes/base/theme.css: false
```

Libraries

```
# In mytheme.info.yml
```

```
libraries-override:
```

```
  system/maintenance:
```

```
    css:
```

```
      theme:
```

```
        # Replace the System module's maintenance CSS file with a custom one.
```

```
        css/system.maintenance.css: css/maintenance.css
```

```
        # The first path is relative to the System module. The second path is
```

```
        # relative to my theme.
```

Libraries

```
# In mytheme.info.yml
```

```
libraries-override:
```

```
  # Replace Classy's messages library with a custom one.
```

```
  classy/messages:
```

```
    mytheme/messages
```

```
  # Remove Classy's search results library completely.
```

```
  classy/search-results: false
```

Libraries

Libraries are attached in three basic ways

- Globally (all pages) from the `mytheme.info.yml` file
- In a Twig file using `attach_library()`
- Conditionally in PHP (preprocess functions)

Libraries

Attaching globally from mytheme.info.yml

```
libraries:
```

- core/jquery
- mytheme/global-styling

Libraries

Attaching in a Twig file

```
{{ attach_library('mytheme/node-styling') }}
```

Libraries

Attaching conditionally in PHP

```
<?php
```

```
function mytheme_preprocess_page(&$variables) {  
    $variables['#attached']['library'][] = 'mytheme/some-other-library';  
}
```

```
?>
```

Twig Basics

Set a variable

```
{% set var = 'Hello, World!' %}
```

Set an array

```
{% set array_var = ['foo', 'bar'] %}
```

Print a variable

```
{{ var }}
```

Print output from a function

```
{{ dump() }}
```

Commenting

```
{# My comments go here. #}
```


Twig Basics

Magic!

TWIG VARIABLE RESOLUTION

```
<p>{{ user.name }}</p>
```

```
$user['name']
```

```
$user->name
```

```
$user->name()
```

```
$user->getName()
```

```
$user->isName()
```

```
NULL
```

*Pick the first
one that exists*



Twig Basics

Filters are applied using a pipe (|) and then the filter name

```
{% set var = 'Hello, World!' %}  
{{ var|upper }}
```

Output

```
HELLO, WORLD!
```

The `without` filter replaces using `hide()` in Drupal 7

```
{{ content|without('comments', 'links') }}
```

More on Drupal 8's filters

<https://www.drupal.org/node/2357633>

Twig Basics

Control structures

```
{% if var %}
    <h2>{{ var }}</h2>
{% else %}
    <p>Nothing here.</p>
{% endif %}

{% set items = ['foo', 'bar', 'baz'] %}
{% for item in items %}
    <p>{{ item }}</p>
{% endfor %}
```

<http://twig.sensiolabs.org/doc/tags/index.html>

Twig Basics

Template inheritance

blah.html.twig

```
...
{% block content %}
    {{ content }}
{% endblock %}
...
```

blah--my-custom-thing.html.twig

```
{% extends "blah.html.twig" %}

{# You only override the 'block' #}
{% block content %}
    <p>I do whatever I want inside the block.</p>
    {{ content|without('something') }}
{% endblock %}
```

API changes

- All theme functions converted to Twig
- 'template' is now the default for theme implementations
- Most CSS classes removed from preprocess

New 'Attributes' class added

```
<div{{ attributes }}>
<div{{ attributes
  .addClass('blue')
  .removeClass('red')
  .setAttribute('id', 'slideshow1') }}>
```

Put it all together

Example from Classy's block.html.twig template

```
{%
  set classes = [
    'block',
    'block-' ~ configuration.provider|clean_class,
    'block-' ~ plugin_id|clean_class,
  ]
%}
<div{{ attributes.addClass(classes) }}>
  {{ title_prefix }}
  {% if label %}
    <h2{{ title_attributes }}>{{ label }}</h2>
  {% endif %}
  {{ title_suffix }}
  {% block content %}
    {{ content }}
  {% endblock %}
</div>
```

New Base Themes

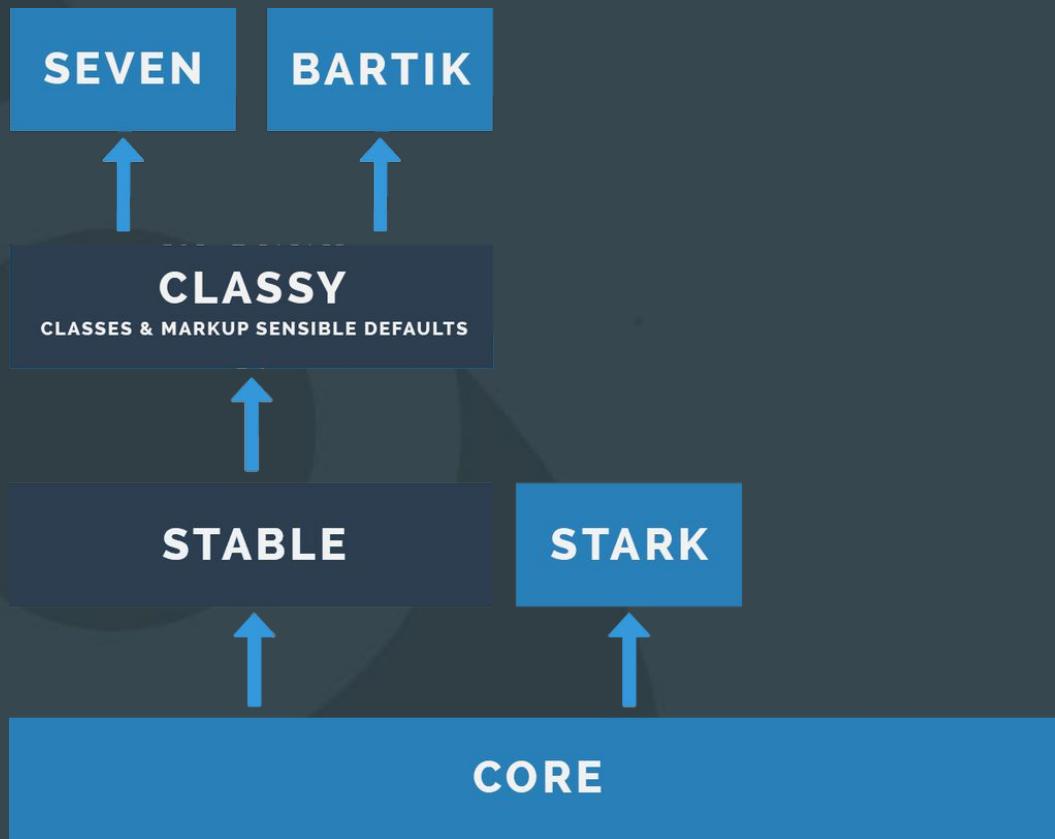
Stable

- A copy of all core templates and CSS files
- Serves as a backwards compatibility layer
- Used by default if not base theme is set
- Set `base theme = false` in `mytheme.info.yml` to disable (**not recommended!**)

Classy

- Modified template files and CSS to provide sensible defaults
- CSS classes added that do not exist in core or Stable

New Base Themes



Questions?