Sassy Drupal

Styling Drupal from CSS to Sass

Cesar Jimenez

Web Developer

The Goal

What is the goal of this talk?

Goal:

To understand and take away one of the following:

- Basic Concepts of Sass
- How to use Sass in Drupal
- Understand how Compass helps us improve workflow with its basic features.

What is Sass?

Syntactically Awesome Stylesheets

What is Sass?

Sass is CSS,
plus nested rules, variables, mixins, and more, all in a concise, readable syntax.

- Hampton Catlin
Originally wrote Sass

What is Sass?

Preprocessing:

Lets you use features that don't exist in CSS yet

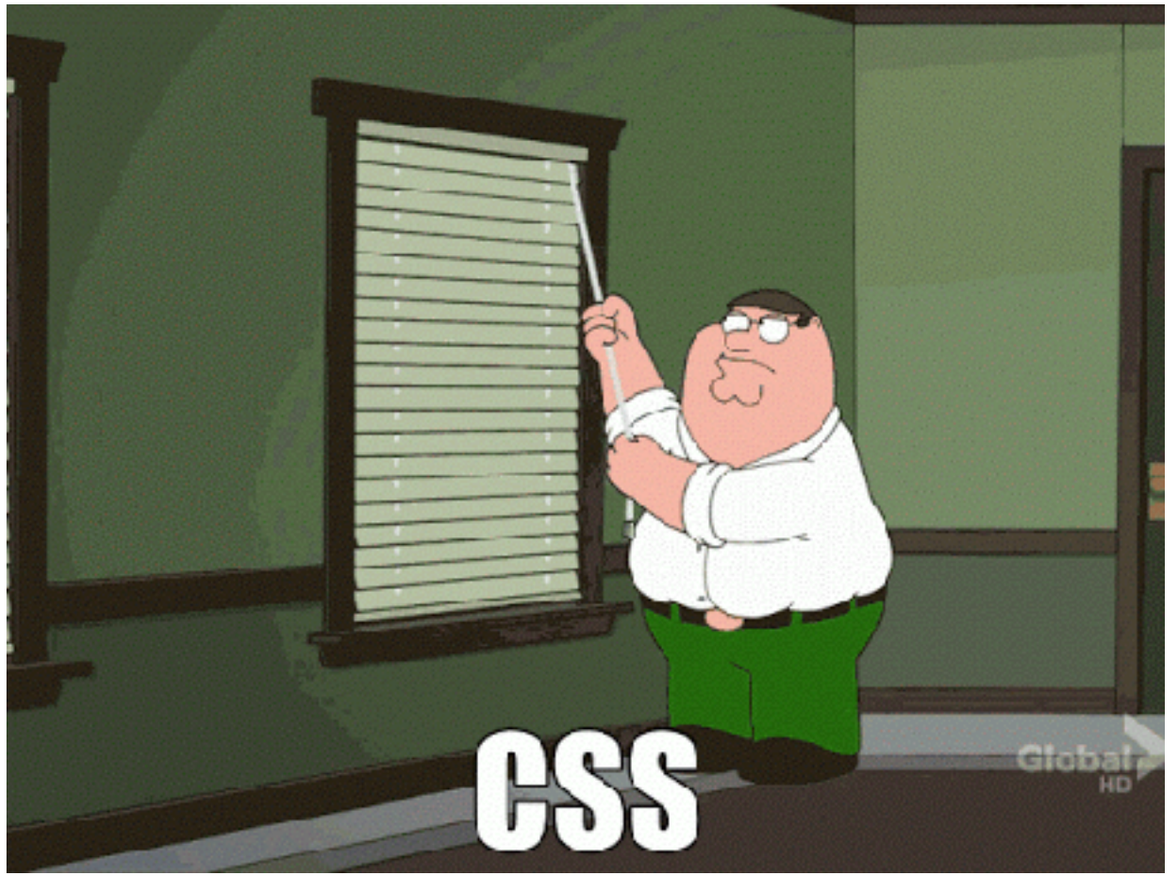
The What

What is the problem Sass solves?

The problem Sass solves

The trouble with CSS:

- Messy
- Repetitive code
- Difficult to maintain



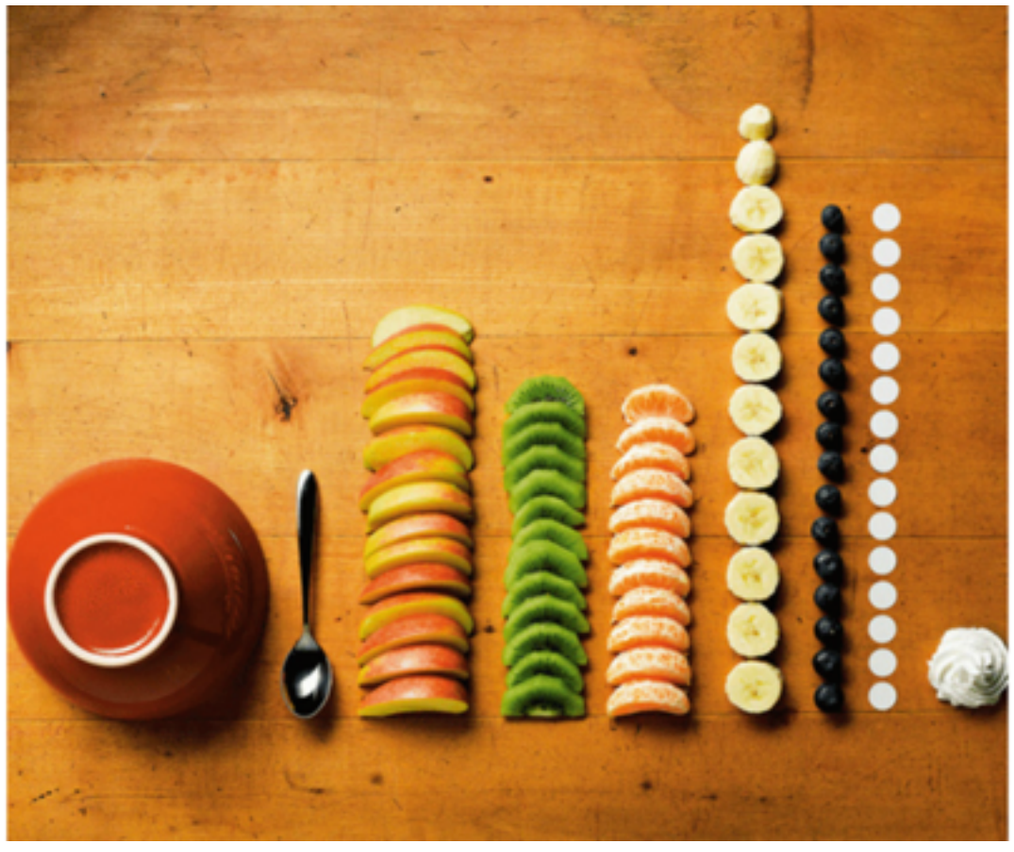
The How

How does Sass solve this?

Let's talk messiness!

CSS on its own is very fun, but stylesheets do get large and messy

Let’s talk messiness!



.scss file crafted by amazing developer

.css file given to the browser

Let’s talk repetition!

How to stay DRY:

Variables

Store information that you want to reuse throughout your stylesheet.

Variables

```
1 // Variables defined
2 $font: Helvetica;
3 $primary-color: #333;
4
5 // Variables used.
6 body {
7   font: $font;
8   color: $primary-color;
9 }
```

Operators

Math in your CSS

Sass has a handful of standard math operators like +, -, *, /, and %.

Operators

```scss
// Variables defined
$small-margin: 10px;
$medium-margin: $small-margin + 5px;
$large-margin: $medium-margin * 2;


// Variables used.
div {
  margin: $medium-margin;
}
```

Mixins

Groups of CSS declarations that you want to reuse throughout your site.

Mixins

Some things just need to be repeated.

The Dude doesn’t approve…

Example of Mixins

```
.box {
    border-radius: 4px;
    border-top-right-radius: 8px;
    border-bottom-left-radius: 8px;
 }
.button {
    border-radius: 4px;
    border-top-right-radius: 8px;
    border-bottom-left-radius: 8px;
    background: #345;
}
```

Mixins 101

Declaring a mixins with @mixin + name

```
@mixin roundy {
    border-radius: 4px;
    border-top-right-radius: 8px;
    border-bottom-left-radius: 8px;
}
```

Mixins 101

Use the mixins with @include, like so:

```
.box {
    @include roundy;
}
```

```
.button {
    @include roundy;
    background: #345;
}
```

Let’s tie it together!

```scss
@mixin roundy {
  border-radius: 4px;
  border-top-right-radius: 8px;
  border-bottom-left-radius: 8px;
}

.box {
  @include roundy;
}

.button {
  @include roundy;
  background: #345;
}
```

I know what you're thinking….

The Power of Mixins

You could even pass in values
to make it more dynamic.

Sounds a little out there for CSS?

The Power of Mixins

Mixins + Arguments = Amazing CSS Functions

```
@mixin roundy($radius-size) {
    border-radius: $radius-size;
    border-top-right-radius: $radius-size * 2;
    border-bottom-left-radius: $radius-size * 2;
}
```

The Power of Mixins

Use @include with arguments, like so:

```
.box {
    @include roundy(4px);
}
```

```
.button {
    @include roundy(4px);
    background: #345;
}
```

Let's tie it all together!

```scss
// Variables
$radius-default: 4px;
$brand-color: #456;

// Mixin
@mixin roundy($radius-size) {
  border-radius: $radius-size;
  border-top-right-radius: $radius-size * 2;
  border-bottom-left-radius: $radius-size * 2;
}

// Implemetation of Sass Amazingness!
.box {
  @include roundy($radius-default);
}

.button {
  @include roundy($radius-default);
  background: $brand-color;
}
```

Bonus: Mixin takeaway

Some things are tedious to write in CSS

…like vendor prefixes.

Bonus: Mixin takeaway

```scss
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
  -moz-border-radius: $radius;
  -ms-border-radius: $radius;
  border-radius: $radius
}

.div {
  @include border-radius(10px);
}
```

The Dude is happy!

Extend / Inheritance

One of the most useful features of Sass:

@extend

Share a set of CSS properties from one selector to another.

Example - Extend

```
.message {
  border: 1px solid black;
  padding: 10px;
  color: #333;
}

.success {
  @extend .message;
  border-color: green;
}

.error {
  @extend .message;
  border-color: red;
}
```

Extend vs. Mixin

@extend

Share a class' properties with another class

@mixin

Declare properties and reuse them throughout with @include

Example - Extend

```
.message {
  border: 1px solid black;
  padding: 10px;
  color: #333;
}

.success {
  @extend .message;
  border-color: green;
}

.error {
  @extend .message;
  border-color: red;
}
```

Example - Mixin

```scss
// Variables
$radius-default: 4px;
$brand-color: #456;

// Mixin
@mixin roundy($radius-size) {
  border-radius: $radius-size;
  border-top-right-radius: $radius-size * 2;
  border-bottom-left-radius: $radius-size * 2;
}

// Implemetation of Sass Amazingness!
.box {
  @include roundy($radius-default);
}

.button {
  @include roundy($radius-default);
  background: $brand-color;
}
```

Let’s talk maintenance!

Maintenance

Maintaining Sass with:

- Nested rules
- Importing
- Partials

Nested rules

Much like HTML, Sass provides a clear nested and visual hierarchy

Nested rules - Sass file

```
// Nested rules
nav {

  ul {
    margin: 0;
    padding: 0;
    list-style-type: none;
  }

  li {
    display: inline-block;
  }

  a {
    display: block;
    text-decoration: none;
  }

}
```

Nested rules - CSS output!

```
// CSS output!

nav  ul {
    margin: 0;
    padding: 0;
    list-style-type: none;
}

nav  li {
    display: inline-block;
}

nav a {
    display: block;
    text-decoration: none;
}
```

Partials

You can create partial Sass files with an underscore.

_concerts.scss

Import

You can use your partials with:

@import

Importing partials!

```scss
// _reset.scss

html,
body,
ul,
ol {
   margin: 0;
   padding: 0;
}
```

```scss
/* base.scss */
@import 'reset';

body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

The Where

So, where does Drupal come in?

Drupal + Sass

You decide

Two Routes for Sassiness

You decide:

- Ruby
- PHP-based Sass processing

Which route to take?

Why Drupal Route?

- Don’t want to install Sass gems?
- You are hardcore Drupal
- Base-theme mixing/variables on sub-themes!

Drupal Modules

Sass/SCSS module (https://www.drupal.org/project/sass)

- Is currently seeking a co-maintainers

Drupal Modules

Compass stylesheet tool
(https://www.drupal.org/project/compass)

- You still need Ruby and PHP in your environment.
- Compass config would need to be managed.
- Versions and permission could become an issue, but why sweat it.

Drupal Modules

Sassy module (https://www.drupal.org/project/sassy)

- Has three dependencies
  - Prepro (http://www.drupal.org/project/prepro),
  - Libraries (http://www.drupal.org/project/libraries)
  - PHPSass (http://github.com/richthegeek/phpsass/downloads).

Which route to take?

Ruby Route

- Note: Prerequisite Ruby gems
- Installing it all with: `gem install compass`
- Need config.rb file

Ruby Route

Simple config.rb file

```ruby
# config.rb

preferred_syntax = :sass
http_path = '/'
css_dir = 'assets/css'
sass_dir = 'assets/sass'
images_dir = 'assets/images'
relative_assets = true
line_comments = true

# output_style = :expanded
output_style = :compressed
```

Ruby Route

`cd` into your theme directory and:

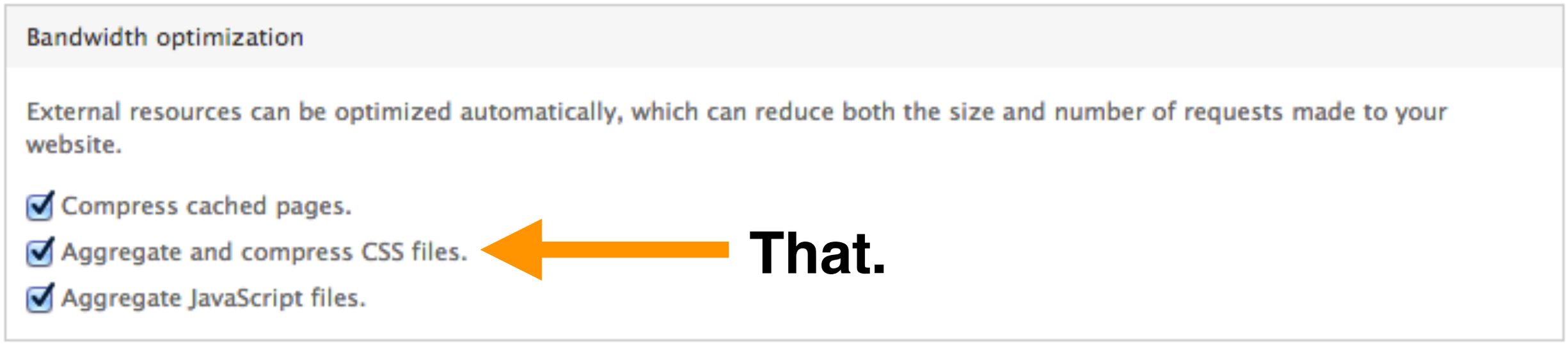
`compass compile` <—-> compiles css 😊

`compass clean` <—-> deletes css 😳

`compass watch` <—-> watch & compiles css 😎

Pro Tip

Good idea to click the radio button with the words "Aggregate and compress CSS files"



under config/development/performance

Pro Tip

Added benefits are:

- Cache-busting the file name will be replaced with “css_v5hdg7naldfnalgueklaa.css”
- <link> instead of <style>@import</style> (older versions of IE are slower with the latter)
- Drupal replaces url (../images/button.png) with url(fullpath/images/button.png) which is great if you have assets on a CDN or subdomain.

Okay, okay…

What about debugging?

SASS Debugging

For the times when things don’t go as planned

SASS Debugging

It's simple.

Change one line in your config.rb file.

SASS Debugging

One line change:

```
output_style = :expanded
```

Emits *.sass “stack trace” for each style

SASS Debugging

```
/* line 379, ../../sass/bootstrap_3.3.1/bootstrap/_normalize.scss */
fieldset {
  border: 1px solid #c0c0c0;
  margin: 0 2px;
  padding: 0.35em 0.625em 0.75em;
}

/* line 390, ../../sass/bootstrap_3.3.1/bootstrap/_normalize.scss */
legend {
  border: 0;
  padding: 0;
}

/* line 399, ../../sass/bootstrap_3.3.1/bootstrap/_normalize.scss */
textarea {
  overflow: auto;
}

/* line 408, ../../sass/bootstrap_3.3.1/bootstrap/_normalize.scss */
optgroup {
  font-weight: bold;
}
```

What now?

When used properly and intelligently,
Sass can be a massive assist in creating websites.

- Go out and play with Sass.
- Look into @if @else statements
- Have fun and thank you!

Q & A

Cesar Jimenez
Web Developer
Metal Toad
cesar.jimenez@metaltoad.com
@cesar_r_jimenez