

Do Sass Me!

An Introduction to **S**yntactically **A**wesome **S**tyle **S**heets

What exactly is Sass?

- Sass is what's known as a **preprocessor**, a program that takes a higher level language and compiles it down to a lower level language
- In this case, we write Sass to be compiled to CSS

Why use Sass?

- Sass allows you to use advanced CSS features that haven't been released yet "officially" or may never be released
 - variables
 - nesting
 - mixins
 - inheritance
 - less-type syntax (woo DRYness!)
- Technically, the variant of Sass that we're learning is called "SCSS" - there is an older one just called "Sass" with different syntax

Sassmeister

- sassmeister.com
- A convenient online GUI to get started with SASS

Variables

- Variables are used to store commonly repeated values in Sass files
- To create a variable, use a \$ sign

```
$default-blue: #0E0EFF;
```

```
body {  
  background-color: $default-blue;  
}
```

Variables - Use Cases

- Imagine your designer creates a color scheme to use for your website. Instead of sampling their designs to get the required hex codes each time, you could have a `colors.scss` file with variables for each of these commonly used colors
- Instead of arbitrarily deciding the widths of various elements of your website, you could set a `$unit` variable and use math to base all of your element sizes on this unit. The same might go for font sizes

Nesting

- Whereas HTML markup normally has nesting of elements that makes a ton of sense, CSS doesn't really have this kind of structure
- With Sass, we can nest our styles!

Nesting

With CSS

```
nav ul{  
  margin: 0;  
}
```

```
nav li{  
  display: inline-block;  
}
```


Nesting

With Sass

```
nav {  
  ul {  
    margin: 0;  
  }  
  li {  
    display: inline-block;  
  }  
}
```

Exercise

- Create a simple navigation menu using Sass
 - Use nesting as discussed in our example
 - The font-size of the links in the menu should be set using a predefined variable

Installing Sass

- Sass is packaged in a Ruby gem, so install it just like you would any other gem from the command line

```
$ gem install sass
```

- Once Sass is installed, you have to set it up to watch the directory your .scss files are in so they can be compiled down to CSS

```
$ sass --watch stylesheets:css
```

- Create a new folder for this project with a directory structure like this:
 - index.html
 - stylesheets
 - main.scss
- Attempt to run the Sass watcher to compile this file into a CSS file, then link the CSS file in the <head> section of your HTML file
- Preview the HTML file in the browser to make sure the process worked - if your background is orange, it did!

Syntax Highlighting

- Install Package Control if you need to (<https://packagecontrol.io>)
- Tools -> Command Palette -> Package Control: Install Package -> Sass

Partials

- Partials are Sass files with small snippets of style, typically repeated often, that you'd like to include elsewhere in your project
- Partials are saved with an underscore:

`_partial.scss`

`_button.scss`

import

- To include a partial inside of one of your files, use the import directive

```
@import 'partial'  
@import 'button'
```

- Notice that you don't have to include the _ or .scss

Exercise

- Create a directory with the following structure:
 - index.html
 - stylesheets
 - main.scss
- Run `sass --watch stylesheets:css` to compile your CSS
- Copy your code from Sassmeister into the HTML and CSS files
- Once you're done, move the navigation menu into a partial file that is imported in your main Sass file

Mixin - Sass Functions

- A mixin is basically a function that you write in Sass, typically to make cross-browser compatibility easier

```
@mixin border-radius($radius) {  
    -webkit-border-radius: $radius;  
    -moz-border-radius: $radius;  
    -ms-border-radius: $radius;  
    -o-border-radius: $radius;  
    border-radius: $radius;  
}  
.box { @include border-radius(10px); }
```

Exercise

- Create a Sass mixin called `black` that takes an opacity as a value and sets the background color to black with the chosen opacity
- Do the same thing for `white`

@extend

@extend allows you to easily bring in the styles from an already declared class into a new class

```
.alert {  
  background-color: #010101;  
  width: 100px;  
}
```

```
/*make a new class with alert's styles and more! */  
.notice {  
  @extend .alert;  
  color: orange;  
}
```

Talk nerdy to me

- Sass also handles mathematical operations
- Just embed math directly wherever you need it

```
$unit: 10px;
```

```
.article-image {  
  width: $unit*10;  
}
```

Comments

```
/* This kind of comment, since it uses  
the normal CSS comment syntax, will appear in the rendered output */
```

```
// This kind of comment is a SASS feature  
// and won't be put inside of CSS output
```

Exercise

- Create a website from the ground up for a fictional business that uses Sass
- Try to make your code as efficient as possible and make use of every single thing we've learned today!

Useful Resources

- <http://thesassway.com/>
- <http://sass-lang.com>