Intro to Sass

Part I

Download Class 1 Files:

erindepew.github.io/gdi-sass-intro/pdfs/class1

Welcome!

Girl Develop It is here to provide affordable and accessible programs to learn software through mentorship and hands-on instruction.

- We are here for you!
- Every question is important
- Help each other
- Have fun

Welcome!

I'm Erin Depew

- Front-end developer at Adobe
- Graphic designer turned programmer
- Four years experience with programming, specializing in functional programming and UI development

Welcome!

Tell us about yourself.

- Who are you?
- What do you hope to get out of the class?

Variables

Without Sass:

```
header {
background-color: #f90000;
  color: #fff;
}
header a { color: #fff; }
```

With Sass Variables:

```
$brandColor: #f90000;
$accentColor: #fff;
header {
  background-color: $brandColor;
  color: $accentColor;
}
header a { color: $accentColor; }
```

Nesting

Without Sass:

```
$brandColor: #f90000;
$accentColor: #fff;
header {
  background-color: $brandColor;
  color: $accentColor;
  }
header a { color: $accentColor; }
```

With Sass Nesting:

```
$brandColor: #f90000;
$accentColor: #fff;
header {
  background-color: $brandColor;
  color: $accentColor;
  a {
    color: $accentColor;
  }}
```

Mixins

Sass Mixins:

```
$brandColor: #f90000;
$accentColor: #fff;
@mixin default-button {
  width: 100%;
  display: block;
  text-align: center;
}
.button--fancy {
  @include default-button;
  background: $brandColor;
  color: $accentColor;
}
```

Mixins

Sass Mixins Compiled:

```
.button--fancy {
  background: #f90000;
  color: #fff;
  width: 100%;
  display: block;
  text-align: center;
  }
```

Tools

- Browser: Chrome or Firefox
- Development Toolkit: Chrome Inspector or Firebug for Firefox
- Command Line
    - Terminal for Mac (Find in Applications> Utilities)
    - Git for Windows (see how to install)
- Text Editor: Sublime Text

Terms

- Ruby: A programming language
- Preprocessor: A computer program that modifies data to conform with the input requirements of another program.
- Compile: The act of converting one computer program into another programming language.
- CSS3: The output of Sass. Sass compiles into CSS!

Sass and CSS

- 1. Styles are written in Sass
- 2. Ruby transpiles Sass to CSS
- 3. CSS files are generated

Questions?

Command Line Tips

Up one level:

```
$ cd ../
```

Go to your home directory:

```
$ cd /
```

Go to a specific folder:

```
$ cd Users/cfarman/Sites/gdi-sass
```

List files in a directory:

```
$ ls
```

Command Line Cheat Sheet!

Installing Sass

First, we need to install Ruby:

- Download for Windows: rubyinstaller.org
- Mac users: you're in luck! Mac OS X comes pre-installed with Ruby. Try the following command in Terminal to make sure:

```
ruby -v
```

Installing Sass

Install the Sass gem by using Terminal or Git Bash:

```
gem install sass
```

Setting Up Stylesheets

We need to structure our stylesheets before we can compile them.

- open the "class1-exercises" folder
- open the "practice" folder in Sublime Text
- go ahead and open index.html in a browser

Setting up our stylesheets

- rename CSS files to have a .scss file extension
- structure them inside the stylesheets folder like so:
  - stylesheets/
    - css/ (keep this directory empty for now)
    - scss/
      - font/
      - reset.scss
      - styles.scss
- Update your index.html stylesheet url in <head> to point to your css/ folder.

Compile

We need to compile our Sass files to make the CSS work in the browser.

First, navigate via the command line to your /stylesheets directory in the "practice" folder.

Then type:

```
$ sass --watch scss:css
```

Sass Watch Command

```
$ sass --watch scss:css
```

--watch look for changes to our .scss files, and compile them to css if they have updates

scss is the location of the Sass files

css is the location for the compiled Sass (aka CSS) files

Nesting

Sass input:

```
header {
  color: black;
  nav {
  background: red;
  a { color: white; }
  }
```

CSS output:

```
header { color: black; }
header nav { background: red; }
header nav a { color: white; }
```

Advanced Nesting

Prefixing:

```
p {
  body.no-touch & {
    display: none; // hide the message if not on a touch device
  }
}
```

BEM syntax:

```
.header {
  &__title{
    font-size: 20px;
    color: black;
  }
}
```

Parent Selectors

Sass input:

```
nav {
    background: red;
    a {
      color: white;
      &:hover { text-decoration: underline; }
    }
}
```

CSS output:

```
nav { background: red; }
nav a { color: white; }
nav a:hover { text-decoration: underline; }
```

Let's Develop It!

- Open your styles.scss file
- Rewrite some styles to use nesting and referencing the parent
- Look for selectors that share a common parent HTML element
- Look for hover styles or add some
- There are lots of possible solution, be creative!
- Run the sass --watch command to see your changes in the browser

Break Time!

Stand up and stretch - we'll resume in 5 minutes

Variables

- Colors

```
#2a79af
```

- Font stack styles

```
Georgia, Times, "Times New Roman", serif;
```

- Font sizes

```
1.667em
```

Defining Variables

Define once, use everywhere

```
//define using dollar sign
$brandColor: #f90000;
$mainTextcolor: #fff;
$accentColor: #ccc;
```

```
$brandColor: #f90000; // red
$mainTextcolor: #fff; // white
$accentColor: #ccc; // grey
```

```
header {
  color: $mainTextColor;
  background-color: $brandColor;
}
.content {
  color: $mainTextColor;
  background-color: $accentColor;
}
footer {
  color: $accentColor;
  background-color: $brandColor;
}
```

Let's Develop It

- Create a new Sass stylesheet called _utilities.scss
- Import your new stylesheet into styles.scss by putting the following code at the top of styles.scss:

```
@import "_utilities";
```

- Create new variables in your _utilities.scss file
- Run the sass --watch command to see your changes in the browser

Syntax Highlighting

You can download tools to highlight your Sass properly in Sublime Text:

- Install Package Manager for Sublime Text 2
- Go to Tools > Command Palette. Type "Package Control".
- Click "Install Package"
- Type "Sass" and click the first result
- Click: View > Syntax > Sass

Math Operations

With CSS you have to be explicit about everything, including numbers. With Sass, you can write math to calculate numbers for you:

+	Addition
-	Subtraction
*	Multiplication
/	Division*

*division is special, check the documentation link for why and how

Math Operations

Sass input:

```
$layoutWidth: 960px;
#sidebar {
  width: $layoutWidth/3;
}
```

CSS output:

```
#sidebar {
  width: 320px;
}
```

Math Operations

Sass input:

```
$layoutWidth: 960px;
$defaultPadding: 16px;
#main {
  padding: $defaultPadding;
  width: $layoutWidth - $defaultPadding*2;
}
```

CSS output:

```
#main {
  padding: 16px;
  width: 928px;
}
```

Let’s Develop It

- Write a math expression in Sass to calculate the width of elements in your page layout instead of declaring a number
- Use a variable to represent the result of your calcuation
- Compile to CSS and refresh your index page to see your changes

```
$layoutWidth: 960px;
$navWidth: $layoutWidth/3;
footer {
  width: ($layoutWidth - 20px);
}
```

Color Functions

Lighten function

Sass input:

```
$linkColor: #000;
$linkShadow: lighten(#000, 40%);
a {
  color: $linkColor;
  text-shadow: $linkShadow;
}
```

CSS output:

```
a {
  color: #000;
  text-shadow: #666;
}
```

Color Functions

Darken function

Sass input:

```scss
$background: #ff0000; // red
$text: darken($background,50%);
body {
  color: $text;
  background: $background;
}
```

CSS output:

```css
body {
  color: #990000;
  background: #ff0000;
}
```

Color Functions

Grayscale function

Sass input:

```
$background: #ff0000; // red
$text: darken($background,50%);
body {
  background: grayscale(#f00);
  color: grayscale(darken(#f00, 50%));
}
```

CSS output:

```
body {
  background: #000;
  color: #808080;
}
```

Let's Develop It

- Edit your variables in _utilities.scss file to use color functions
- Refer to the sass-lang.com docs
- Compile to CSS to see your changes
- Bonus: change the color scheme without editing styles.scss!

```
lighten(#000, 20%)
darken(#eee, 30%)
grayscale(#2a79af)
saturate(#2a79af, 40%)
invert(#2a79af)
```

Mixins

One or more style rules that can be reused

Sass input:

```
@mixin dropshadow($text) {
  color: $text;
  text-shadow: 2px 4px lighten($text, 50%);
}
p {
  @include dropshadow(black);
}
```

CSS output:

```
p {
  color: black;
  text-shadow: 2px 4px #808080;
}
```

Let's Develop It

- Add mixins to your _utilities.scss file
- Use the mixins in your styles.scss file

```
@mixin name {
  property: value;
}
@mixin example($argument) {
  property: value;
  property: $argument;
}
```