INTRO TO SASS

CLASS 1

DOWNLOAD CLASS 1 FILES:

cfarm.co/sass1

WELCOME!

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

Some "rules"

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

WELCOME!

I'm Jen Myers

- Designer/Developer/Instructor @ Dev Bootcamp
- I've been making things with HTML/CSS for over ten years.
- I started GDI Columbus and ran GDI Chicago for a little while; now I'm an advisor and sometimes teacher.
- I have a kid. Ask her about her website.

WELCOME!

Tell us about yourself.

- Who are you?
- What do you hope to get out of the class?
- What's something no one in this room knows about you?

WHAT DOES SASS DO?

Variables:

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

WHAT DOES SASS DO?

Nesting:

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

WHAT DOES SASS DO?

Mix-ins:

```
@mixin visually-hidden {
  text-indent: 100%;
  white-space: nowrap;
  overflow: hidden;
}
.image-replace {
  @include visually-hidden;
  background: url(logo.png) no-repeat;
}
```

WHAT DOES SASS DO?

More examples:

- Cross-Browser CSS3 Gradients
- Cross-Browser Box-shadow (drop shadow) effects
- Inner Text Shadow effect
- Animated CSS Progress Bars
- Multiple CSS Animations in a sequence
- Off-Canvas Responsive Layout

TOOLS

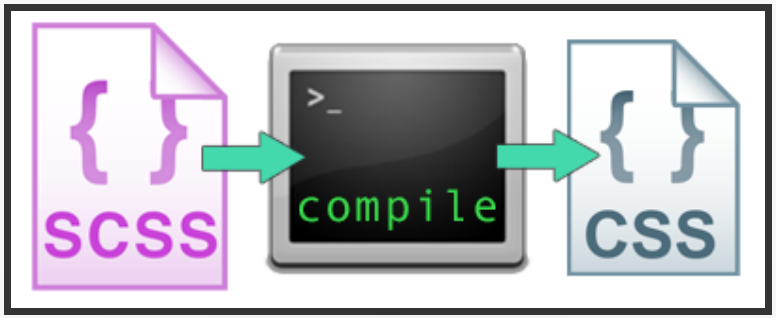
- Browser: Chrome or Firefox
- Development Toolkit: Chrome Inspector or Firebug for Firefox
- Command Line
  - Terminal for Mac (Find in Applications> Utilities)
  - Git for Windows (see how to install)
- Text Editor:
  Sublime Text - Mac or Windows
  sublimetext.com/2

TERMS

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

SASS AND CSS

How Sass compiles into CSS



1. You write Sass (.scss files)
2 .You run a command (sass --watch) on the command line
3. .scss files compile into .css files

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!!! (In class1-exercises folder)

SASS SET UP

You should have installed Sass on your computer prior to this workshop.

Check if Sass is installed:

```
gem list sass
```

You should see something like this:

```
# *** LOCAL GEMS ***
# sass (3.3.4)
```

Raise your hand if Sass is not installed.

If installtion doesn't seem to be working, you can use Sass Meister for now.

SETTING UP OUR 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/
      - empty for now
    - /scss/
      - /font/
      - reset.scss
      - styles.scss
- Update your index.html stylesheet url in <head> to point to your /css/ folder.

COMPILE WITH THE SASS WATCH COMMAND

Check out your index.html file in a browser. Looks funky, yes?

We need to compile our .scss (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
```

COMPILE WITH THE SASS WATCH COMMAND

Let's break this command down - you're going to use it *a lot*!

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

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

The **scss** bit is the folder where our .scss files live. We edit these files only.

The **css** part is the folder where our .css files will be. These files are the compiled output of our Sass, and are only created when we run the above command.

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; }
```

REFERENCING 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, like header, nav, footer, #main
- Look for hover styles, or add some, to practice referencing the parent with &
- There are lots of possible solutions! 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

Sometimes you want to reuse a value for a style - you use them frequently, they're hard to type or remember, such as

- Colors

```
#2a79af
```

- Font stack styles

```
Georgia, Times, "Times New Roman", se
```

- Font sizes

```
1.667em
```

DEFINING VARIABLES

Variables with Sass let us reuse values more than once, while only defining them in one place

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

To create a variable you need a dollar sign before the name of your variable, and a colon: to give it a *value*

Note that in Sass files, you can comment out a line with // two slashes

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

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

KEEP EM TOGETHER

Variables are easy to change if you keep them all in one stylesheet, and update or add to the styles as needed

```
// Let's define some variables
// Colors
$favoriteColor: #2a79af;
$anotherColor: #f05b62;
// Fonts
$favoriteFont: Papyrus, fantasy;
$aPracticalFont: "Trebuchet MS", "Lucida Grande", "Lucida Sans Unicode", "Luc
// Font sizes
$aNiceBigFontSize: 16px;
$finePrint: 10px;
$giantLogo: 36px;
// Margins and Padding
$defaultMargin: 15px;
$defaultPadding: $defaultMargin;
```

LET'S DEVELOP IT

- Create a new Sass stylesheet called _utilities.scss
- Don't forget to put it in the right folder
- Import your new stylesheet into styles.scss by putting the following code at the top of styles.scss:

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

- In your new stylesheet, make some new variables, and base some variables on your existing styles - look for colors, fonts, and size values
- 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

Color functions are built-in to Sass. They let you alter existing color values. This is the lighten function:

Sass input:

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

CSS output:

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

Example of output style

COLOR FUNCTIONS

This is the darken function:

Sass input:

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

CSS output:

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

Example of output style

COLOR FUNCTIONS

This is the 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;
}
```

Example of output style

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 of your website without editing styles.scss!

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

COMMENTS

Traditional CSS comments are downloaded by the user. With Sass, we can specify whether comments are left in the final CSS code or are only visible to the developer:

Sass input:

```
/* Multiline comments will appear
 in the final CSS file */
//This single-line comment won't
a { color: blue; }
```

CSS output:

```
/* Multiline comments will appear
 in the final CSS file */
a { color: blue; }
```

MIXINS

Mixins are really just a collection of one or more styles that are reusable, like variables are reusable values

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;
}
```

Example of output style

MIXINS

These are especially useful for CSS3 rules that need browser prefixes, like gradients.

cfarm.co/gradient

LET'S DEVELOP IT

- Add some mixins to your _utilities.scss file - a good candidate is any style that needs a browser prefix
- Try replacing the footer gradient with a mixin
- Use these mixins in your styles.scss file
- Add different types of comments that describe what your mixins do, and check the final CSS in the browser inspector to see if they appear

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

QUESTIONS?