Web Development 2

Day 3

Agenda

- Sass
- Lab time

About Sass

- Sass stands for:

S yntactically

A wesome

S tyle

S heets

About Sass

- Sass is a CSS preprocessor and scripting language
- Developers write CSS like syntax that is then interpreted by the Sass complier and converted into regular CSS
- Sass is open source, free to use and many implementations exist.
- Sass gives developers script like abilities for their CSS.
- The following are available to CSS developers that use Sass:
  - Variables
  - Mixins (sort of like functions)
  - Functions
  - If/else statements
  - Loops
  - @extend
  - Importing and concatenation of several files into a single file
  - Compression
  - “//” single line comments

Why Use Sass

- It adds programmability to CSS
- Promotes DRY (do not repeat yourself) principals in writing your CSS
  - You can use variables, mixins, functions and extends to write a set of rules once and apply them in multiple locations
- It allows you to break up your CSS into smaller modular files (called partials) and concatenate them into a single CSS file, making development easier
- We can use the quick and easy “//” for single line comments

Where Sass Gives you the Most Benefits

- Larger websites that have a large CSS code base
- Websites that have multiple developers working on the CSS of a website

How Sass Works

SCSS Compilier CSS

Installing Sass

- Many options to get Sass up and running on your computer
  - Use a build system such as Gulp
  - Use a GUI tool such as CodeKit or Prepros
  - Install the Dart version
  - Install the Node.js version (The Node.js version is a bit slower to compile)
- For this class we will run Sass from the Terminal (Mac) or the Command Prompt (Windows) using the Dart version of Sass
- For future classes you will more than likely use Node Sass in combination with a build system
  - More on build systems in an upcoming class
- The Sass syntax is the same regardless of whether you use Dart Sass or Node Sass

Special Note about the “$” Character and the Terminal/Command Prompt

- If you are ever reading a tutorial that covers entering commands into the terminal or a Windows command prompt you will often see the “$” followed by the command
- Never enter the ”$” character at the start of a terminal command
- The “$” is a generic placeholder for your terminal prompt
- Since everyone's prompt is slightly different, people use the “$” character to represent the prompt

Installing Dart Sass - macOS

1. Install Homebrew
   - Visit: https://brew.sh/
   - Follow the install instructions
2. Type the below command into the macOS Terminal App
   - Do not type in the “$”

```
$ brew install sass/sass/sass
```

Installing Dart Sass - macOS

- If you see this error

```
$ Error:
        homebrew-core is a shallow clone.
        homebrew-cask is a shallow clone.
```

- Run the following to updage brew in the command line:

```
$ git -C /usr/local/Homebrew/Library/Taps/homebrew/homebrew-core fetch –unshallow
$ git -C /usr/local/Homebrew/Library/Taps/homebrew/homebrew-cask fetch --unshallow
```

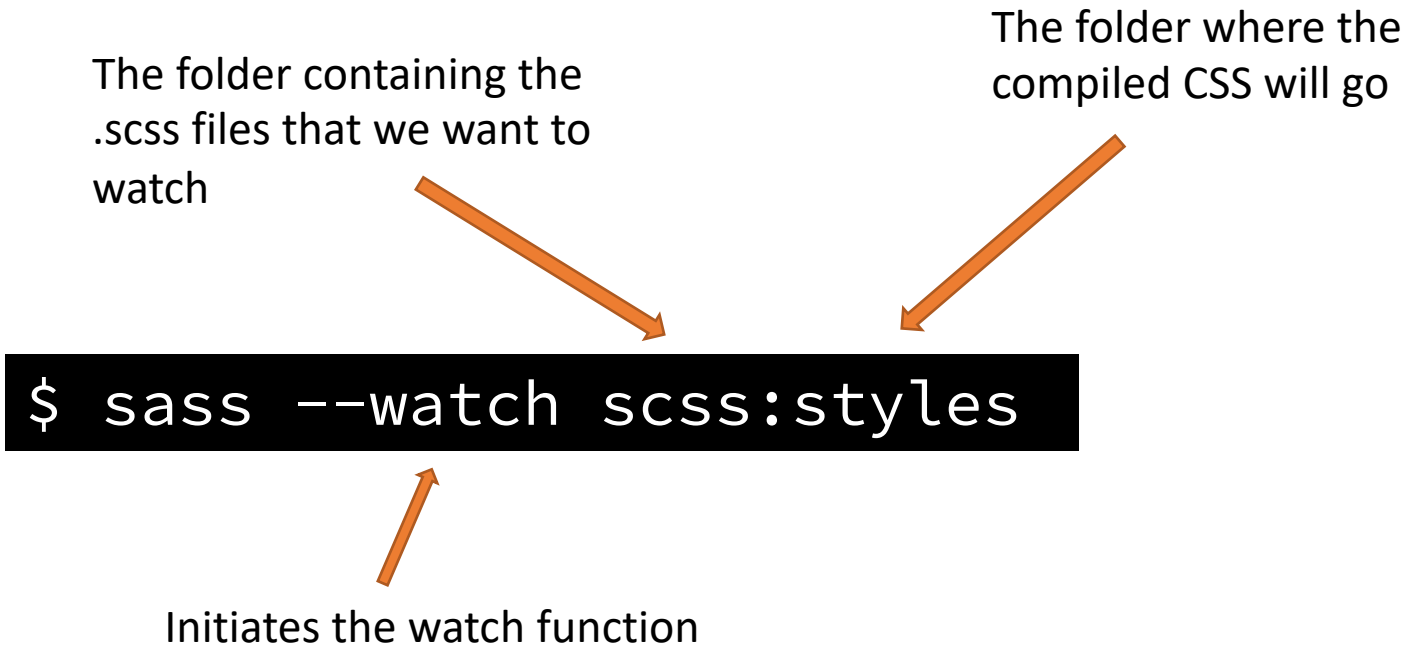
Installing Dart Sass - Windows

1. Install Chocolatey
   - Visit: https://chocolatey.org/
   - Click the “Install Now” button
   - Follow the install instructions
2. Type the below command into Windows PowerShell
   - Do not type in the “$”

```
$ choco install sass
```

Watch Files With Sass

- In order for Sass to compile files you need to tell Sass what folder to watch
- To tell Sass to watch a file or folder of files first navigate to your project folder in the terminal and then type the following:



Your Browser only understands CSS

- A key point to remember is that your browser only reads CSS files, so the link to your stylesheet in your HTML file should be to your compiled CSS file, not to your Sass files
- The Sass compiler will create a ".map" file which some browser's developer tools can read.
- The ".map" file will tell the browser which Sass file and line number a particular CSS rule was compiled from
  - That information is displayed in the browser's developer tools to aid in CSS/Sass development

Don't Edit the Compiled CSS File

- Do not edit the compiled CSS
- If you need to change the CSS, write your changes in Sass and re-compile
- Feel free to look at the compiled CSS to see what Sass is outputting for troubleshooting purposes

Writing Sass

- Sass can be written in two ways
  - Sass syntax
    - Uses indenting to format rules
  - SCSS syntax
    - Written using a syntax that is similar to the syntax used for writing standard CSS
    - For this course we will be using SCSS syntax but your choice of syntax comes down to either personal preference or what your development team is using

Sass Syntax

```
$brand-color: #eee;

p
  font-size: 24px
  color: $brand-color
  line-height: 1.5
```

SCSS Syntax

```
$brand-color: #eee;

p {
  font-size: 24px;
  color: $brand-color;
  line-height: 1.5;
}
```

Sass Features

- Variables
  - Sass variables work slightly differently from CSS custom properties…see next slide for details
- Mixins
- Partials
- Imports
- @extend
- Functions
- Loops
- If/else statements

CSS Custom Properties vs Sass Variables

- Advantages of CSS Custom Properties
  - Native to the browser (no build step required)
  - Can be changed dynamically by JavaScript
  - Can be changed based on @media conditions

- Advantages of Sass Variables
  - Preprocessors render variables as standard CSS property values so can be used in any browser of almost any age (deep browser support)
  - Can be used in more places than CSS custom properties
    - Set as a "property"
    - Set as a "selector"
    - Set as a media query statement

@use vs @import

- @use and @import work similarly in that they both allow you to write smaller partial files and “import” those files into other files, allowing for more modular CSS development
  - How they accomplish this is slightly different
- As of November 2021 the people behind Sass discourage the use of @import
  - You will still see a lot of Sass tutorials utilizing @import
  - Your @import statements still work in Sass today, but that may not be the case going forward, so I encourage you to familiarize yourself with using @use instead of @import in your Sass development
  - For this class we will learn the @use syntax

@use vs @import

- Read the Sass team “Heads up!” statement about using @import here:
  - https://sass-lang.com/documentation/at-rules/import
- Learn about @use here:
  - https://sass-lang.com/documentation/at-rules/use

VS Code – Sass Extension

- Useful extension for Sass in Visual Studio Code
  - SCSS IntelliSense
    - https://marketplace.visualstudio.com/items?itemName=mrmlnc.vscode-scss
    - This extension will auto-complete variables, mixins, functions and extends for a values located across the project
      - Especially useful when using partials

Sass Resources

- Sass web site
  - http://sass-lang.com