Getting Started with Sass

Due: Next class

Grading: 20 points possible

Sass, or Syntactically Awesome Stylesheets, is a CSS preprocessing language that makes creating stylesheets faster and easier to maintain. With Sass, you create Sass stylesheets using a variation of CSS that takes advantage of efficiency features built into the language. Sass then outputs regular CSS that browsers understand to your stylesheet file. Sass features include *nested rules*, *variables*, *mixins*, *partials* and more that extend what CSS is capable of. A good place to get acquainted with Sass is on their Sass Basic page. With Sass you create stylesheets with the file extension ".scss" (there is an older extension ".sass" for an older version of the language), which then gets output as regular css with the ".css" file extension.

There are a few ways to run Sass. Normally, Sass is run with a command line program like Terminal. But this requires a Sass installation to your computer's operating system (which in turn requires the programming language Ruby). Alternatively, you can run Sass using a free program called Scout, which contains Sass and is easy to install. This is what we will use. With this exercise you will setup Scout to run Sass and create Sass files containing color and typography variable information. This will become the initial CSS for Breakroom.

Exercise details:

- Setup Sass workflow with Scout
- Create master Sass file
- Create Sass partials for the following:
  - Variables
  - Typography
  - Patterns
- Integrate Google web font
- Output Sass file to CSS

Directions

Add classes to markup

You will start off by adding a few new classes to your Breakroom HTML.

1. Launch a web editor and open, *index.html*. Add a *class* attribute to the *header* tag with value

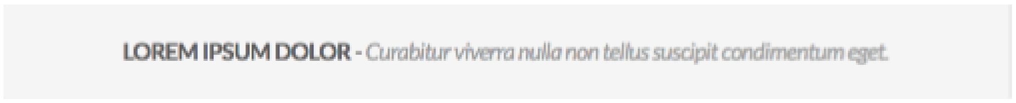
"header".

2. Add a *class* attribute to the *footer* tag with value "footer".

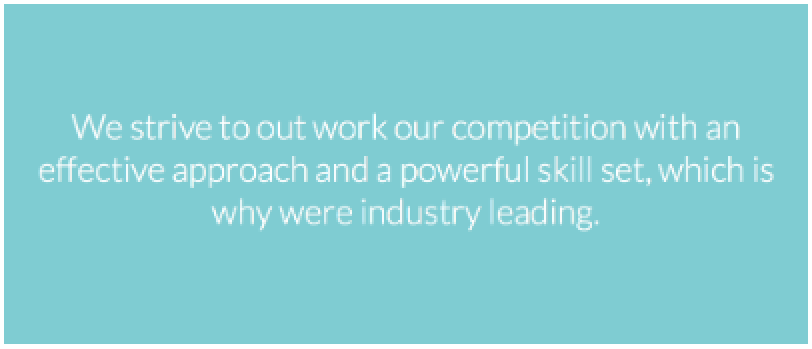
3. Add a *class* attribute to the first *section* tag with value "footer-top".

4. Add a *class* attribute to the second *section* tag with value "footer-bottom".

5. Add ".well" class to the *section* tag in *home.html* with "LOREM IPSUM ..." text. Also, update the *h2* tag to an *h3*.


LOREM IPSUM DOLOR - *Curabitur viverra nulla non tellus suscipit condimentum eget.*

6. Add ".message" class to all *section* tags with turquoise background color.

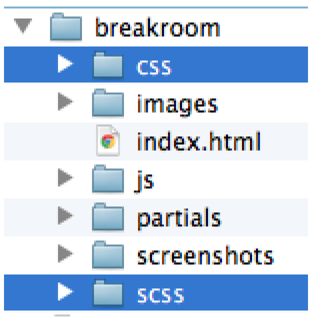

We strive to out work our competition with an effective approach and a powerful skill set, which is why were industry leading.

7. Add ".lead" class to 1st *paragraph* tag after page h2s (*about.html*, *services.html*, *contact.html*).

```
    <h2>About Us</h2>
    <p class="lead">Paragraph.lead - Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed
    do eiusmod tempor incididunt.</p>
```

Create local scss and css directories

8. Go to your local *breakroom* project folder and create two new sub-folders: "css" and "scss".


breakroom
css
images
index.html
js
partials
screenshots
scss

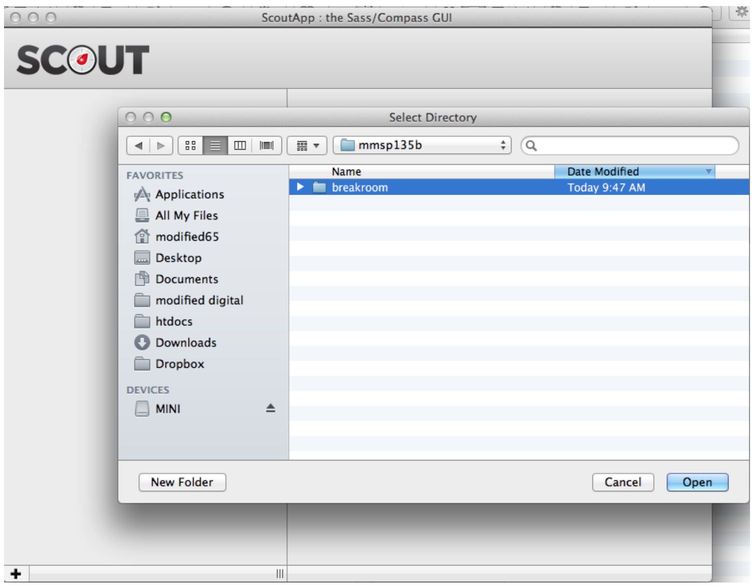
Configure Scout for Sass workflow

9. Look for the Scout application icon and launch it.

Configure project directories for Scout

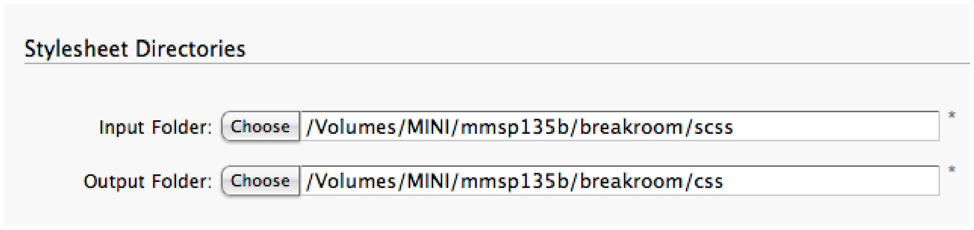
Next, will define the location where your source Sass files will live and what folder Scout will output the converted CSS to.

10. With Scout launched, click the little "**+**" icon in the lower left-hand corner and click the **Open** button once you have selected *breakroom*.



11. Next, under *Stylesheet Directories*, configure the input and output folder fields. The **input**

folder should point to the *scss* directory you created**,** and the **output folder should point to the *css* folder** you created.

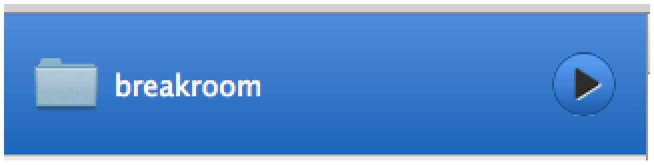

Stylesheet Directories

Input Folder: Choose /Volumes/MINI/mmsp135b/breakroom/scss

Output Folder: Choose /Volumes/MINI/mmsp135b/breakroom/css

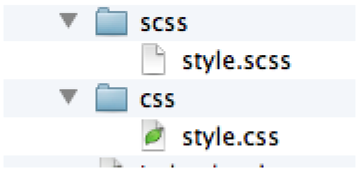
You can leave the other settings as is for now.

12. Next, click the start button to have Scout watch for new and updated scss files output as css.



Create master Sass file

13. Launch a web editor and create a new blank file called “style.scss”. Save it in your local *scss* folder. With Scout running in the background, a blank “style.css” should get outputted to your *css* folder automagically! Switch to the finder window to confirm this.



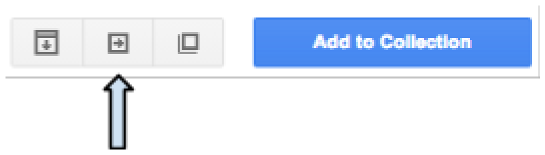
Add “Lato” web font

Switching gears for a moment, the web font “Lato” is used for headings throughout this project and the first thing you will do is reference this web font at the top of the *style.scss*.

14. Launch a browser and go to Google Web Fonts: **http://www.google.com/webfonts**

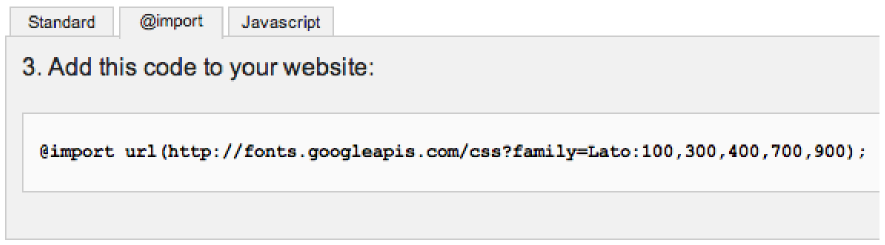
15. In the left-hand column, enter “Lato” into the search form.

16. Next, click the **Quick Use** button.



17. Next, under “Choose the styles you want”, check “Thin 100, Light 300, Normal 400, Bold 700, and Ultra-Bold 900”.

18. Next, scroll down and at step 3, “Add this code to your website”, click the “@import” tab.



3. Add this code to your website:

```
@import url(http://fonts.googleapis.com/css?family=Lato:100,300,400,700,900);
```

19. Next, with *style.scss* open, and paste the import code below the at the top of the stylesheet.

Create Sass variables partial

Partials allow you to break apart your stylesheet into sub-files with specific areas of focus. Sass then merges partials into one file when outputting. The first partial you will create will define a series of variables that contain color and typography values for Breakroom.

20. Create a new blank file and save it as “_variables.scss” in your local *scss* folder (the leading underscore identifies this file as a Sass partial).

There are 6 basic colors in the Breakroom design pallette. Reference the screenshots to pick up the hex (or rgb) values.

21. Add the following variable names to *style.scss* below the import syntax. The first color value is provided as an example.

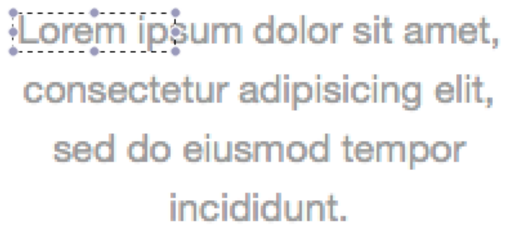
```
// Colors
```

```
$white: #fff;
$black: ;
$turquoise: ;
$gray: ; // paragraph text
$darkGray: ; // headings, footer top background color
$lightGray: ; // paragraph text
```

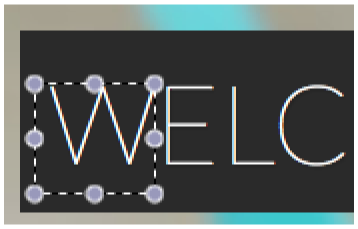
22. Next, after the color variables, define variables for typography. These will cover base font information as well as headings 1-3.

    For h1-h3 font sizes, reference the *breakroom-index-1024.png* screenshot to measure (also, see below). For font size units of measurement, you can use *pixels* or *ems*. Note that when using *ems*, values will be relative to the font-size set for the body or the containing element font-size, whichever is applicable.

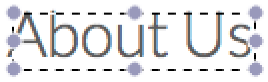
```
// Typography
$baseFontFamily: "Helvetica Neue", Helvetica, Arial, sans-serif;
$baseFontSize: ; // measure
$baseLineHeight: inherit;

$headingsFontFamily: "Lato", sans-serif;

$h1FontSize: ; // measure
$h1FontWeight: 100;

$h2FontSize: ; // measure
$h2FontWeight: 300;

$h3FontSize: ; // measure
```


Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt.

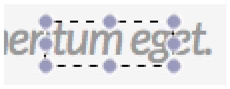
Base font size (breakroom-index-1024.png)



Heading 1 (breakroom-index-1024.png)



Heading 2 (breakroom-about-1024.png)



Heading 3 (breakroom-index-1024.png)

Create typography partial

23. Create a new blank file and save it as "_typography.scss" in your local *scss* folder.

24. Within *_typography.scss*, create CSS rules that use the color and typography variables instead of static values. Add the following:

```
body {
    font-family: $baseFontFamily;
    font-size: $baseFontSize;
    line-height: $baseLineHeight;
    color: $lightGray;
    background-color: $white;
}

h1 {
    font-family: $headingsFontFamily;
    font-size: $h1FontSize;
    font-weight: $h1FontWeight;
    color: $white;
    background-color: $black;
}
```

```
h2 {
    font-family: $headingsFontFamily;
    font-size: $h2FontSize;
    font-weight: $h2FontWeight;
    color: $darkGray;
}

h3 {
    font-family: $headingsFontFamily;
    font-size: $h3FontSize;
    font-weight: $h3FontWeight;
    color: $gray;
}
a {
    color: inherit;
    &:hover,
    &:focus {
      text-decoration: ; // remove underline
    }
}
```

This last rule takes advantage of Sass's nested rules feature. The ampersand is required when

nesting pseudo selectors. The above rule will output as the following in *style.css*:

```
a {
    color: inherit;
}
a:hover {
    text-decoration: ; // remove underline
}
a:focus {
    text-decoration: ; // remove underline
}
```

Create patterns partial

The patterns partial is where reusable classes specific to Breakroom will get placed.

25. Create a new blank file and save it as "_patterns.scss" in your local *scss* folder.

26. Within *_patterns.scss*, add the following CSS rules associated with the classes you recently added to the Breakroom HTML.

```
.header {
    background-color: $black;
}

.footer-top {
    background-color: $darkGray;
    color: $white;

    h2 {
            color: $turquoise;
    }
}

.footer-bottom {
    background-color: $black;
}

.message {
    background: $turquoise;
    h2 {
            color: $white;
    }
}

.lead {
    color: $turquoise;
    text-transform: capitalize;
}
```

Update master Sass file to include partials

27. Next, open *style.scss* and add references to the partials under the web font link. The syntax is "@import" followed by the name of the partial in quotes, i.e. "variables". You don't include the

trailing underscore or the ".scss" extension.

```
@import url(http://fonts.googleapis.com/css?family=Lato:100,300,400,700,900);
@import "variables";
@import "typography";
@import "patterns";
```

28. Save your changes. Your collection of partials should be merged as one *style.css* file in your *css* folder. Open *style.css* to confirm.

Add link to local stylesheet

To apply the CSS to Breakroom, you just need to link to the local stylesheet.

29. Open *index.html* and add a link to the *style.css* within the css folder. Add the link **after** the link to Font Awesome.

```
<link
href="https://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css"
rel="stylesheet">
<link href="css/style.css" rel="stylesheet">
```

Test using local server

30. Use Brackets preview to launch Chrome in a local server environment or by running an alternative local server.

Transfer project folder to web host

31. Use an FTP client like Fugu or Filezilla to transfer your project folder to your remote account. You don't need to upload your *scss* folder. That can stay on your local drive.

Test after uploading

32. Launch a web browser, go to your project URL, and make sure that your loads as expected.

Notify instructor when exercise is complete

33. Email me the URL to your remote *breakroom* project at chris.clark@mail.ccsf.edu.

For next class

- What is OOCSS? (Overview of Object Oriented CSS)
- Architecting CSS (Speaker Deck presentation)
- Level Up Tuts: Sass Tutorials (nice collection of Sass tutorial videos)