Project 4: Sass style Guide

Sections of this Guide:

- **How to approach this project** includes detailed guidance to help you think about how to organize your code, project and files.
- **How to succeed at this project** lists the grading requirements for the project, with hints, links to course videos to refresh your memory and helpful resources.

How to Approach this Project

For this fifth project, you will use Sass to add responsive mobile first styles to the supplied index.html file so that it matches the provided mockups when loaded in a browser.

❏ **Download the project source files from the Sass Style Guide project instructions page in your Techdegree curriculum.**

❏ **Set up a new GitHub repo and push the project files to it.**

  ❏ *Related video:* Share Your Projects with GitHub

❏ **Install Sass if you haven't done so already.**

  ❏ *Related video:* Installing and Using Sass
  ❏ *Related video:* Installing and Setting up Sass

❏ **Create your Sass directory:** *details below in the "How to Succeed at this Project" section***.**

❏ **Run `sass --watch scss:css` in the console.**

  ❏ *Related video:* Compiling Sass to CSS
  ❏ *Related video:* Compiling a Directory of Sass Files
  ❏ *Related video:* The Watch Command

At this point, a good approach is to begin by creating the variables, placeholders, mixins, and functions that you know you are going to use. Then start writing Sass to style the mobile layout. Once the mobile layout matches the mockup, move on to styling the desktop layout. More variables, placeholders, mixins, and functions can always be added as you code the styles for this project.

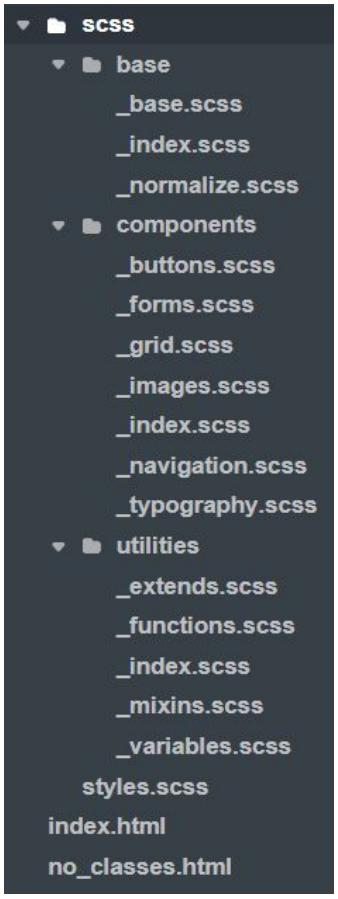
- ❏ **Things to keep in mind as you go:**
  - ❏ **Components Folder:** The comment at the top of the index.html file contains a list of the classes you will use to style this project. Use these classes when adding styles to the files in the components subfolder. **Tip:** You can use tag selectors in the components folder as well, but only as child selectors. Example: .class a {}.
  - ❏ **Base Folder:** Only use the universal selector, `*`, `ul` tag, and `p` tags in the `_base.scss` file.
  - ❏ **Utilities Folder:** Variables, placeholders, mixins, and functions should go in the Utilities folder.
  - ❏ **Tip**: Avoid combining selectors from two different categories. For example, avoid using `.grid__col--8 .link` to style the link in the typography section on the page, as it makes the link styles dependent on what grid column they're in.
  - ❏ **Tip:** Use only nested media queries: Nested Media Queries
  - ❏ **Bugs and Errors:** Review Sass and CSS debugging videos if you have trouble styling an element, run into layout bugs, or recieve a Sass error.
    - ❏ *Related video:* Debugging Errors in Sass
    - ❏ *Related video:* Handling Error with @error and @warn
    - ❏ *Related video:* Debugging in the Browser with Source Maps
    - ❏ *Related video:* Debugging CSS with Chrome Dev Tools
- ❏ **Double check everything, validate your files, request an informal review in Slack, and then submit.**

How to succeed at this project

Here are the things you need to do pass this project. Make sure you complete them **before** you turn in your project.

❏ Project folders

- ❏ Create a `scss` folder, and inside of that, create a `styles.scss` file and three sub-folders, `base` , `components` , and `utilities`.
- ❏ Inside each of the three sub-folder, create an `_index.html` "partial" along with all rest of the "partial" files as shown in the image here: —>
- ❏ Use `@import` statements to import "partials" into the `styles.scss` and all partials.
  - ❏ *Related video:* Importing partials - ***related content at the 2:13 minute mark.***



❏ Variables

- ❏ Place the `_variables.scss` file is in the utilities folder.
- ❏ Create at least one variable for fonts, breakpoints, and colors.
    - ❏ *Related video:* Declaring and Using Variables
    - ❏ *Related video:* Naming Variables
    - ❏ *Related video:* Color Variables
    - ❏ *Related video:* Font and Asset Variables
    - ❏ *Related video:* Media Queries and Breakpoint Variables

❏ Mixins

- ❏ Create mixins for the following:
    - ❏ Media Queries
    - ❏ Flexbox settings
    - ❏ *Related video:* Introducing Mixins
    - ❏ *Related video:* Creating Powerful Mixins
    - ❏ *Related video:* Smarter Mixins with Arguments
    - ❏ *Related video:* Reusing code with Mixins
    - ❏ *Related video:* Mixins for Text Properties
    - ❏ *Related video:* Media Query Mixins
    - ❏ *Related video:* CSS Flexbox Layout

❏ Grid

- ❏ Create a Sass @for loop to iterate over each grid__col class.
- ❏ Inside of loop, create a formula for making adjustable widths based on the number of the col. For instance, a grid__col--12 would span (roughly) 12/12 or 100% of the total width of the row, while a grid__col--3 would span (roughly) 3/12 or 25% of the total width of the row.
- ❏ Related video: **Grid Configuration**
- ❏ Related video: **Generating Column**
- ❏ Related video: **Column Layout**
- ❏ Related video: **Building the Grid Container**

❏ Design

- ❏ When viewed in a browser, index.html should match overall design of mobile and desktop mockups.

- ❏ When hooked up to your output CSS, and loaded in a browser, no_classes.html matches appearance of no_classes_mockup.png file.
- ❏ index.html file has not been altered.

Tip: If you have trouble styling an element, run into layout bugs, or recieve a Sass error, refer to "Things to keep in mind as you go" in the previous section of this doc for related videos.