

Coding & Cocktails CSS Compilers Worksheet

Overview

During the session we'll introduce CSS Compilers and how they can make your styling easier. We'll start our project with normal CSS styling and transform that into a clean and organized stylesheet using Sass! If there is a term you don't know check out our glossary here: bit.ly/CnCgloss



Prep Work

1. Utilize our Tools worksheet (bit.ly/cnctools) to install our recommended tools prior to the session.
2. Another Sublime Text plugin you'll want to have installed for Sass development is called Sass. You can view installation instructions here: bit.ly/cncsass
3. Sass has a Ruby dependency this means we need Ruby (another programming language) installed to use it.
 - a. Windows based machines please follow the instructions here: rubyinstaller.org
 - b. Macs, you're lucky, you've already got it installed.

4. Open your command line terminal (Git Bash on Windows, iTerm2 on Macs) and type `gem install sass`

Output should look similar to :

```
➔ ~ gem install sass
Fetching: sass-3.4.21.gem (100%)
Successfully installed sass-3.4.21
Parsing documentation for sass-3.4.21
Installing ri documentation for sass-3.4.21
Done installing documentation for sass after 6 seconds
1 gem installed
```

5. Check that you've got everything installed by typing `sass -v`
Output should look like this:

```
➔ ~ sass -v
Sass 3.4.21 (Selective Steve)
```

Project

In order to compare the differences between CSS and SASS we're going to work through styling a sample page. First we'll add CSS styling to get a bit of practice and make it pretty. Then we'll translate the CSS into Sass utilizing variables, mixins and nesting to make our development easier! **Note: If you need a refresher on Git refer to the March worksheet here: bit.ly/CnCvers. A refresher on the Command line can be found here: bit.ly/cmdln**

Part 1: Styling with CSS

1. Choose a color palette from colorhunt.co and make note of the color hex values (the #XXXXXX that displays when you hover over the color) to use as the colors in your project. You can also just use the one we selected: colorhunt.co/c/3986
2. Scroll through the list of fonts on google webfonts (google.com/fonts) and choose two fonts that you like and look nice together for your website.
 - a. Click the blue "Add to Collection button" on the right of each one you want to use
 - b. At the bottom of the page click on the "Use" button.
 - c. Leave just the "Normal" box checked and scroll down to step 3 "Add this code to your website:"
 - d. Click on the import tab and copy that code and paste it in the top of your CSS file.
 - e. Or use the fonts we selected: Montserrat & Neuton

```
@import url(https://fonts.googleapis.com/css?family=Montserrat|Neuton);
```

3. Fork the CSSCompilerPractice repository (bit.ly/CnCCmp) to your own account
4. Clone the repository to your computer using Git Bash or iTerm2
5. Open the index.html file in Google Chrome. It doesn't look very nice, does it? Let's add some styling!

6. Open the CSSCompilerPractice folder in Sublime Text. Add CSS to the *styles.css* file to style your page. Reference the CSS cheat sheet below for some of the commonly used CSS properties if you need a refresher or try googling things! If you need some inspiration try to make your page look similar to the image here: bit.ly/CnCLDFS
 - a. Style a horizontal navigation bar (If you need some help see w3schools: bit.ly/NavCSS)
 - b. Style the main “hero” image with the large page heading, a smaller “tagline” and some about text
 - c. Style a product list
 - d. Style the sidebar Customer Favorites content to display on the right side
 - e. Style a footer with copyright and social media links
7. Once your page has been styled with CSS compare with the answer key here: bit.ly/CnCCSSKey. It may not be exact depending on how you’ve chosen to style your page and any creative liberties you’ve taken but will give you some idea on what your CSS might look like.

Take a break and grab another drink! You’ve earned it!

Part 2: Re-Styling with Sass - Re-style your page using Sass. View the Sass documentation here if you want more detailed information about how any pieces of Sass: bit.ly/CnCSassDoc

1. Create a *sass* folder in the *assets* folder
2. In the *sass* folder, create a *styles.scss* file. This will be your main Sass styling file.
3. Copy everything from your *styles.css* file into *styles.scss* (*Hint: use ctrl-a (windows) or cmd-a (macs) to select everything in your file*)
4. Rename your *styles.css* file to *originalstyles.css* as we’ll be overwriting this css file when we compile our Sass and it will be neat to compare the two later.
5. Start with switching out colors you used in your CSS file with variables.
 - a. Create a *_variables.scss* file in your *assets/sass* folder. This will be where all of our variables that we will use will exist.
 - b. At the very top of your *styles.scss* file you’ll need to import the variables file so type in `@import “variables”`
 - c. Add variables to the *_variables.scss* file. Remember these start with a \$ followed by the variable name. Examples: `$color-main: #F9A1BC;` and `$color-accent: #A9EEE6;`
 - d. In the places in your *styles.scss* file where you reference these colors change out the hex value (`#F9A1BC`) with your variable name (`$color-main:`)

Now our styling might look similar to this:

```
h1,h2,h3,h4,h5,h6 {  
    color: $color-headlines;  
}
```

- e. Save the Sass files.
 - f. In the command line type `sass assets/sass/styles.scss assets/css/styles.css`
 - g. Go to Chrome and refresh your page. If you’ve used the same colors it should look exactly the same but now you only have one place to change a color instead of spread throughout your CSS file! If you want to see the “magic” switch one of the color variables maybe to white (`#FFFFFF`), re-run your sass command and then refresh your page.
6. Next, let’s change the font stacks to use variables such as `$font-headers` and `$font-body`
 - a. Follow the same process as above adding variables, setting the values and replacing the font stacks in your css to use the variables you created. Don’t forget to compile!
7. Use nesting to simplify your stylesheet - remember not to go more than 3-4 levels deep though!
 - a. In places where you have parent-child selectors, nest that inside the main class for the parent class or element.

b. For example instead of

```
nav ul {  
    ...  
}  
nav li {  
    ...  
}
```

You might have this:

```
nav {  
    ul {  
        ...  
    }  
    li {  
        ...  
    }  
}
```

That way you don't have to repeat nav in front of the child ul and li elements. Similar to the way you nest your html elements you can now organize your Sass file with nesting.

c. Save the Sass files.

d. In the command line type `sass assets/sass/styles.scss assets/css/styles.css`

e. Go to Chrome and refresh your page. It should still look the same.

8. Let's try a mixin now! We can use one for list styling, let's call it list-types.

a. Create a `_mixins.scss` file in the assets/sass folder

b. Add your mixin to the file - this is adding a "function" to set the three properties list-style, margin and padding anytime this mixin is used in your Sass file:

```
@mixin list-reset() {  
    list-style: none;  
    margin: 0;  
    padding: 0;  
}
```

c. In your styles.scss file replace any styling where you set those three items (likely in your navigation, product list and footer elements) with `@include list-reset();`

d. Save both `_mixins.scss` and `styles.scss`

e. In the command line type `sass assets/sass/styles.scss assets/css/styles.css`

f. Go to Chrome and refresh your page. It should again look the same. Replacing your CSS with Sass may not look very exciting but it reduces duplication in your code and makes it easier for you to change and maintain the code going forward! You will typically just use Sass from the start instead of doing the translation from CSS but we wanted you to see the differences between the two types of styling.

9. Feel free to try out any more of the Sass concepts we discussed tonight using the same process.

10. Compare your initial css file with the generated css file. They should be almost identical.

a. In Sublime text highlight the original styles.css file then use ctrl (windows) or cmd (macs) + click on styles.css

b. Right click on one of the highlighted filenames and choose "Diff Files..."

c. This will open a new tab showing the differences (if any) between the two files.

11. Compare your styles.sass with our answer key here: bit.ly/CnCSassKey. It may not be exact but will give you an idea on what your Sass may look like.

Congratulations! You've just styled a webpage using a CSS Compiler!


Homework

The more you practice, the more comfortable you'll feel! Take a look at the following tutorials

1. Walk through this tutorial by 1stWebDesigner.com to get more Sass practice: bit.ly/SassTut (Note: if the pictures don't display appropriately in the tutorial you can click on where they should be showing to display them)
2. For more practice and a focus only on the Sass without worrying about your html try this tutorial from Scotch.io: bit.ly/SassTut3

CSS Cheat Sheet

This cheat sheet shares some commonly used CSS properties. If you want to know how to do something googling it is usually very helpful often taking you to stack overflow or w3schools. Try searching google for `css margin` to get a better understanding of the margin property. Another place to look is the documentation on the mozilla developer network pages here: bit.ly/MDNDOC

Property	Description	Value	Example
margin 	Adds spacing around an element outside the border (refer to the box model diagram to the left) Can separate into margin-top, margin-right, margin-bottom margin-left	Any pixel number value	<pre>p { margin: 10px; } p { margin: 0 auto; }</pre>
padding	Add spacing between the element content and the element border (refer to the box model diagram above)	Any pixel number value	<pre>p { padding: 10px 0 15px 20px }</pre>
border	Specifies the border styling	Width: any pixel number value Color: hex color value Style: solid dashed dotted etc.	<pre>p { border: 2px solid blue; }</pre>
color	Sets the font color	Hex color values or color names	<pre>p { color: #ffffff; }</pre>
text-align	Aligns text according to value	left right center justify	<pre>p { text-align: center; }</pre>
text-decoration	Add decoration to text	none underline	<pre>a { text-decoration: none; }</pre>
font-family	Set the font family (or typeface) of the text	Any font name	<pre>p { font-family: Neuton, "Times New Roman", Times, serif; }</pre>

font-size	Set the size of the text	Numeric value in px, em or rem	h1 { font-size: 4em; }
font-style	Sets the text style	Normal Italic oblique	h4 { font-style: italic; }
font-weight	Set the weight of the text	normal bold bolder etc.	h3 { Font-weight: bold }
background-color	Sets the color of the selected elements background	Any hex color value or color name	body { Background-color: #ffffff; }
background-image	Set the background of the element to the image supplied	Image url	.hero { background-image: url('images/image.png'); }
background-repeat	Specifies whether or not a background image should be repeated	repeat repeat-x repeat-y no-repeat	body { background-repeat: no-repeat; }
background-position	Sets the starting position of a background image	center center left top right bottom etc	.hero { background-position: right center; }
list-style-type	Determines the bullet style on the list	circle square decimal disc upper-roman lower-alpha none	ul { list-style-type: none; }
float	Specifies whether an element should float	Right Left Center none	img { float: left; }
clear	Does not allow floating to the specified side of an element	Right Left both	.footer { clear: both; }