IN4MATX 133: User Interface Software

Lecture 21:
SASS and Styling in Ionic

Professor Daniel A. Epstein
TA Jamshir Goorabian
TA Simion Padurean
Class notes

- Quiz 4 (Tuesday) will cover November 5th’s lecture (Guest lecture by Josh Tanenbaum) through today

- A2 grades will be posted after class
  - Sorry for the delay

- Quiz 3 grades will hopefully be posted early next week

- Next week’s office hours schedule is rearranged to account for Thanksgiving
  - Check the calendar
Today’s goals

By the end of today, you should be able to...

- Explain why we might use a preprocessor like SASS for CSS
- Use SASS variables, mixins, nesting, and operators to simplify CSS
- Style Ionic components and override platform-specific styles
Same page, different stylesheets

Welcome to My Homepage
Use the menu to select different Stylesheets

- Stylesheet 1
- Stylesheet 2
- Stylesheet 3
- Stylesheet 4
- No Stylesheet

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links:
Stylesheet 1, Stylesheet 2, Stylesheet 3, Stylesheet 4.

No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet:
No Stylesheet.

Side-Bar
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliqua erat volutpat.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliqua erat volutpat.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliqua erat volutpat.

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links:
Stylesheet 1, Stylesheet 2, Stylesheet 3, Stylesheet 4.

No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet:
No Stylesheet.

Side-Bar
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliqua erat volutpat.

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links:
Stylesheet 1, Stylesheet 2, Stylesheet 3, Stylesheet 4.

No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet:
No Stylesheet.

Side-Bar
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliqua erat volutpat.

https://www.w3schools.com/css/demo_default.htm
CSS syntax

- **Selectors** specify which elements a **rule** applies to

- **Rules** specify what **values** to assign to different formatting **properties**

/* CSS Pseudocode */

```css
selector {
    property: value;
    property: value;
    ...  
}
```

One rule, many properties
Writing plain CSS

• Violates the “Don’t Repeat Yourself” principle of coding

• Many times we’re writing the same snippets of code for frequently used declarations
Example: fonts

- What if I want to switch from Lato to some other font?
- What if I want to make everything larger?
- I could have structured my CSS more efficiently, but at the core, it’s inflexible
- For example, I could have set Lato to be the default font for cite

```css
cite > .series {
  font-family: 'Lato', sans-serif;
}

cite > a {
  font-family: 'Lato', sans-serif;
  font-size: 0.8em;
}

cite > .authors {
  font-family: 'ChaparralPro', serif;
  font-size: 1.1em;
}

cite > .title {
  font-family: 'Lato', sans-serif;
  font-weight: 700;
}
```
CSS preprocessors

• “Let you abstract key design elements, use logic, and write less code”

• Three widely used ones: SASS, Less, Stylus

• They all do pretty much the same thing
  
  • Ionic uses SASS by default
CSS preprocessors

Major features

• Variables
• Mixins
• Nesting
• Operators
Variables

- Using variables with CSS preprocessors makes it easy to update colors, fonts, or other values throughout your entire stylesheet.

//Sass Variables
$primary: #CC5533;
$font-base: 12px;

//Less Variables
@primary: #CC5533;
@font-base: 12px;

//Stylus Variables
primary = #CC5533
font-base = 12px
Variables

• As of 2016, variables are supported in plain old CSS

• Many frameworks use these variables instead of preprocessor variables

• Including Ionic (new in 4!)

• But preprocessors offer a lot more functionality

/*Declaring a variable*/
element {
  --main-bg-color: brown;
}

/*Using the variable*/
element {
  background-color: var(--main-bg-color);
}

https://www.caniuse.com/#search=css%20variables
Mixins

- Mixins can share a whole collection of CSS rules throughout a stylesheet

```css
@.mixin border-radius($radius) {
  -webkit-border-radius: $radius;
  -moz-border-radius: $radius;
  -ms-border-radius: $radius;
  border-radius: $radius;
}

.box { @include border-radius(10px); }
```
Nesting

- You can nest selectors with preprocessors
- This means you can easily organize an entire hierarchy of selectors, including child elements

```css
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
  li { display: inline-block; }
  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
  &:hover {
    text-decoration: underline;
  }
}
```
Question

Which SASS nesting is equivalent to these plain CSS rules?

A
```css
ul {
  font-weight: 700;
  list-style-type: square;
}
li {
  color: blue;
}
```

B
```css
ul {
  font-weight: 500;
  list-style-type: circle;
}
li {
  color: red;
}
```

C
```css
ul {
  font-weight: 500;
  list-style-type: circle;
}
li {
  color: red;
}
```

D
```css
ul {
  font-weight: 500;
  list-style-type: circle;
}
li {
  color: red;
}
```

E
```css
li {
  color: red;
}
ul {
  font-weight: 500;
  list-style-type: circle;
}
```
Operators

- Like most programming languages, CSS preprocessors can do math!
- This is especially great for setting a fixed value in a variable, like a font-size or padding, and then modifying it as you go along

```css
$container = 100%;

article[role="main"] {
  float: left;
  width: 600px / 960px * $container;
}
```
Digging into SASS
SASS

Syntactically Awesome Style Sheets

• “SASS is the most mature, stable, and powerful professional grade CSS extension language in the world.”

• “SASS boasts more features and abilities than any other CSS extension language out there”

• It’s on their website, so it must be true!

http://sass-lang.com/
SASS

- File extension: .scss
- SASS is a superset of CSS
  - You can write any CSS in a SCSS document
- SCSS is transpiled to CSS
  - Just like TypeScript is transpiled to JavaScript

FIG 1: Sass converts its own "power syntax" to plain old CSS.
SCSS Syntax

- Looks very much like regular CSS
- Rules apply to a selector and are made in brackets
- Each rule ends with a semicolon
- SCSS adds variables, mixins, etc.

```scss
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```
SCSS Syntax

- There’s another style, called “sass syntax”, which looks more like python

- It’s older, uses the .sass extension

- Why is .scss better?
  - It’s a superset of CSS, rather than another syntax
How do I actually use a CSS preprocessor like SASS?
Installing SASS

- `npm install -g sass`
  - This version is written in JavaScript, which is slower
  - But it’s fine for the size of projects we’re working with
- `choco install sass`
- `brew install sass/sass/sass`
  - 3 times to make super duper sure
  - (just kidding, it’s how HomeBrew designates projects)

https://sass-lang.com/install
Manually transpiling

- You can use SASS with a plain-old HTML page!
- It just needs to be transpiled to CSS before getting loaded
Manually transpiling

● Transpile one file:
  • `sass input.scss output.css`

● Watch one file for changes:
  • `sass --watch input.scss:output.scss`

● Watch a whole directory of SASS files:
  • `sass --watch path/sass-directory`
Automatic transpiling

- A lot of frameworks will automatically transpile .scss files when they build and run

- Angular and Ionic can include .scss files for every component and secretly transpile them to .css

- A preprocessor is specified when the app is first created
Thoughts on CSS preprocessors

- Preprocessor functionality is slowly getting added to the CSS standard
  - CSS now supports variables, for example
- Does this mean that preprocessors will soon be obsolete?
  - Maybe. Or maybe they’ll evolve, adding other kinds of new and better features
  - Transpiling languages are a great way to show the value of new features
  - And if they catch on enough, they get added into the standard
  - Who knows, maybe JavaScript will add typing from TypeScript
Styling in Ionic
Ionic variables

- Two types of variables:
  - Global variables
  - Component variables

- Three files:
  - app/global.scss
  - app/theme/variables.scss
  - app/[component]/[component].scss
Ionic global variables

- Defined in app/theme/variables.scss
- Used for defining platform colors, fonts, margins, etc.
- There are ~100 different global variables, but the defaults are pretty good

```css
/** primary **/
:root {
  --ion-color-primary: #3880ff;
  --ion-color-secondary: #0cd1e8;
  --ion-font-family: 'Arial';
  --ion-margin: 16px;
}
```

https://beta.ionicframework.com/docs/theming/advanced/
Ionic global variables

• Global variables can be accessed manually, like other CSS variables

• Ionic provides utilities for accessing global color variables in HTML

• This somewhat violates code separation principles... but the syntax is cleaner

```css
/*SCSS or CSS*/
ion-button {
    color: var(--ion-color-primary);
}

<!--HTML-->
<ion-button color="primary"></ion-button>
```
Color palette

- Ionic's color generator can help you see what your scheme will look like in a built-out app

- May be useful in conjunction with a color palette tool:
  - [https://color.adobe.com/](https://color.adobe.com/)
  - [http://paletton.com/](http://paletton.com/)

- It will generate a file which can be copy/pasted into app/theme/variables.scss

[https://beta.ionicframework.com/docs/theming/color-generator](https://beta.ionicframework.com/docs/theming/color-generator)
Component variables

- Each component has specific variables for defining its style
- These variables can be edited where the component is created (e.g., the parent component)

CSS Custom Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--background</td>
<td>Background of the button</td>
</tr>
<tr>
<td>--background-activated</td>
<td>Background of the button when activated</td>
</tr>
<tr>
<td>--background-focused</td>
<td>Background of the button when focused</td>
</tr>
<tr>
<td>--border-color</td>
<td>Border color of the button</td>
</tr>
<tr>
<td>--border-radius</td>
<td>Border radius of the button</td>
</tr>
<tr>
<td>--border-style</td>
<td>Border style of the button</td>
</tr>
<tr>
<td>--border-width</td>
<td>Border width of the button</td>
</tr>
<tr>
<td>--box-shadow</td>
<td>Box shadow of the button</td>
</tr>
<tr>
<td>--color</td>
<td>Text color of the button</td>
</tr>
<tr>
<td>--color-activated</td>
<td>Text color of the button when activated</td>
</tr>
<tr>
<td>--color-focused</td>
<td>Text color of the button when focused</td>
</tr>
<tr>
<td>--height</td>
<td>Height of the button</td>
</tr>
<tr>
<td>--margin-bottom</td>
<td>Margin bottom of the button</td>
</tr>
</tbody>
</table>

https://beta.ionicframework.com/docs/api/button
Component variables

<!--component.html-->  
<ion-button id="testButton">Test Button</ion-button>

/*component.scss*/
#testButton {
  --border-width: 5px;
  --border-style: solid;
  --border-color: black;
}

testButton

CSS Custom Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--background</td>
<td>Background of the button</td>
</tr>
<tr>
<td>--background-activated</td>
<td>Background of the button when activated</td>
</tr>
<tr>
<td>--background-focused</td>
<td>Background of the button when focused</td>
</tr>
<tr>
<td>--border-color</td>
<td>Border color of the button</td>
</tr>
<tr>
<td>--border-radius</td>
<td>Border radius of the button</td>
</tr>
<tr>
<td>--border-style</td>
<td>Border style of the button</td>
</tr>
<tr>
<td>--border-width</td>
<td>Border width of the button</td>
</tr>
<tr>
<td>--box-shadow</td>
<td>Box shadow of the button</td>
</tr>
<tr>
<td>--color</td>
<td>Text color of the button</td>
</tr>
<tr>
<td>--color-activated</td>
<td>Text color of the button when activated</td>
</tr>
<tr>
<td>--color-focused</td>
<td>Text color of the button when focused</td>
</tr>
<tr>
<td>--height</td>
<td>Height of the button</td>
</tr>
<tr>
<td>--margin-bottom</td>
<td>Margin bottom of the button</td>
</tr>
</tbody>
</table>

https://beta.ionicframework.com/docs/api/button
Platform styles

- Each platform has a mode, either \texttt{ios} (iOS) or \texttt{md} (Android)

- Components and global styles can be adjusted for each mode

```scss
/* in global.scss */
.ios {
  --ion-background-color: #222;
}
.md {
  --ion-background-color: #220;
}
```
Today’s goals

By the end of today, you should be able to...

- Explain why we might use a preprocessor like SASS for CSS
- Use SASS variables, mixins, nesting, and operators to simplify CSS
- Style Ionic components and override platform-specific styles
IN4MATX 133: User Interface Software

Lecture 21: SASS and Styling in Ionic

Professor Daniel A. Epstein
TA Jamshir Goorabian
TA Simion Padurean