



SCORE 2.0 UI

Ben Cato & Emily Lord



## What's New?

- Markup (HTML) is significantly lighter
- SCORE components inherit default styles via Sass/LESS
- Class selections are available to ***all*** SCORE components!

# Course Overview

01

The Value of  
SCORE

02

Sitecore Demo

03

File Structure

04

Where To Start

05

Bootstrap  
(HTML/CSS/JS)

06

SCORE HTML

07

Styling and Inheritance  
with Sass

08

SCORE Components

09

SCORE 2.0  
Migration

# The Value of SCORE

01



# BRAINjOCKS™ SCORE™ Benefits



Reusable components

Portability

Saves Time and Money



# BRAINjOCKS™ SCORE™ Markup

SCORE components are categorized as:

- Container
- Content
- Navigation
- Social
  - ★ New



# BRAINjOCKS™ SCORE™ Markup

SCORE components provide a fixed set of HTML based upon Bootstrap

- HTML is generated from inserting component(s) in Sitecore
- HTML cannot change for SCORE components
- Custom components use custom HTML



# BRAINjOCKS™ SCORE™ Markup

Markup *is portable* and can be manipulated by front-end developer

- Styles for HTML are easily inherited via Sass
- Container components (i.e. the grid) can easily be adjusted via Sass
- HTML is not completely bound to any specific framework
  - ★ Components that require JS are an exception

# Bootstrap Markup

Markup *is not portable* or fluid when used with components

- Front-end dev has no control over column sizes, etc.
- Column sizes are set in HTML for the component and is difficult to change
- Project is committed to a 12-column grid

02

## Sitecore Demo

Connecting SCORE markup  
and SCORE components

# 03

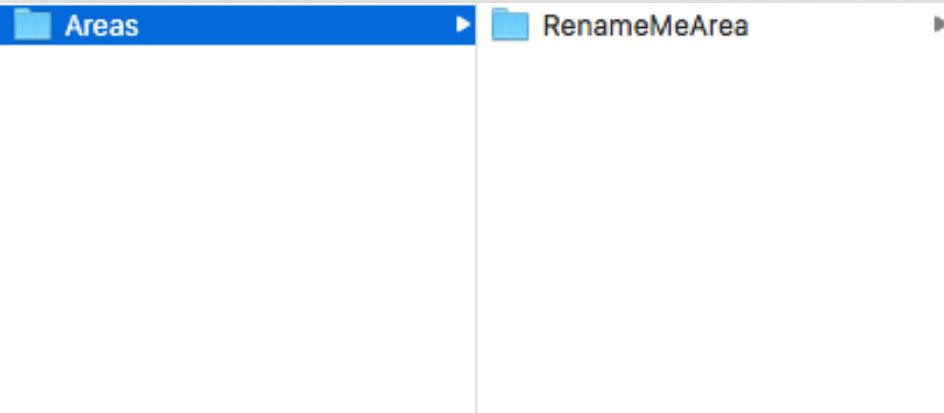
## Understanding SCORE's File Structure

Organization of project files,  
Sass files, and the HTML  
sandbox

# File Structure

## “Areas” Scaffolding

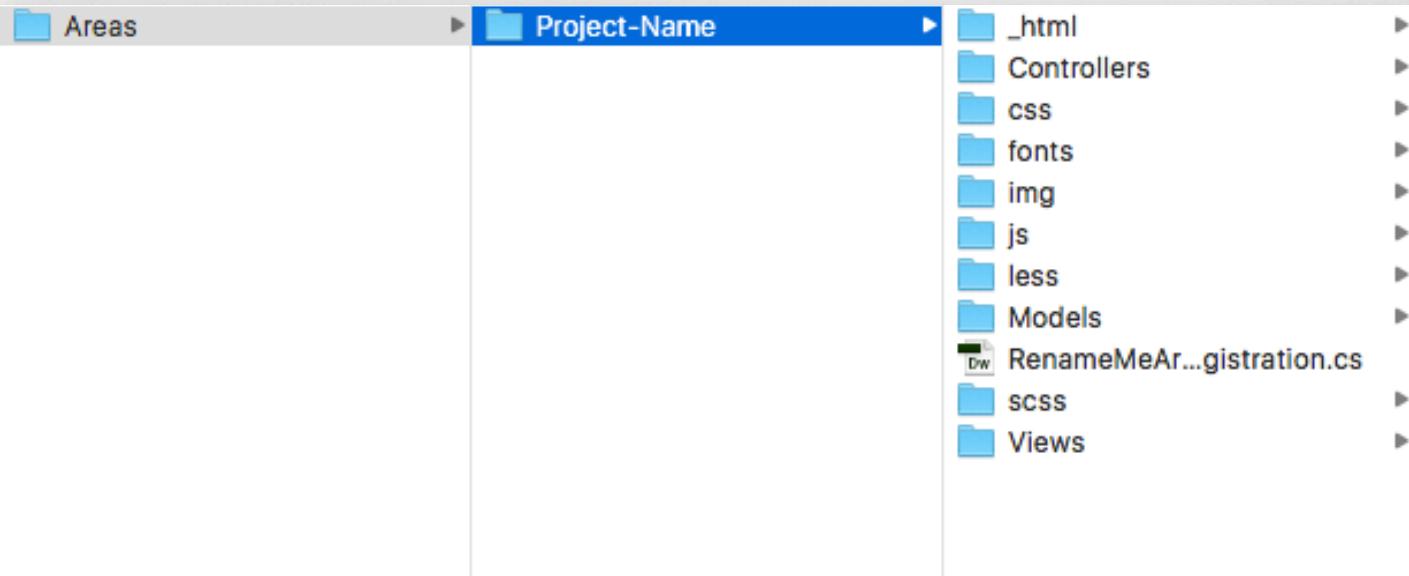
- All project files are *shared* between Sitecore and UI developers



# File Structure

“RenameMeArea” Folder

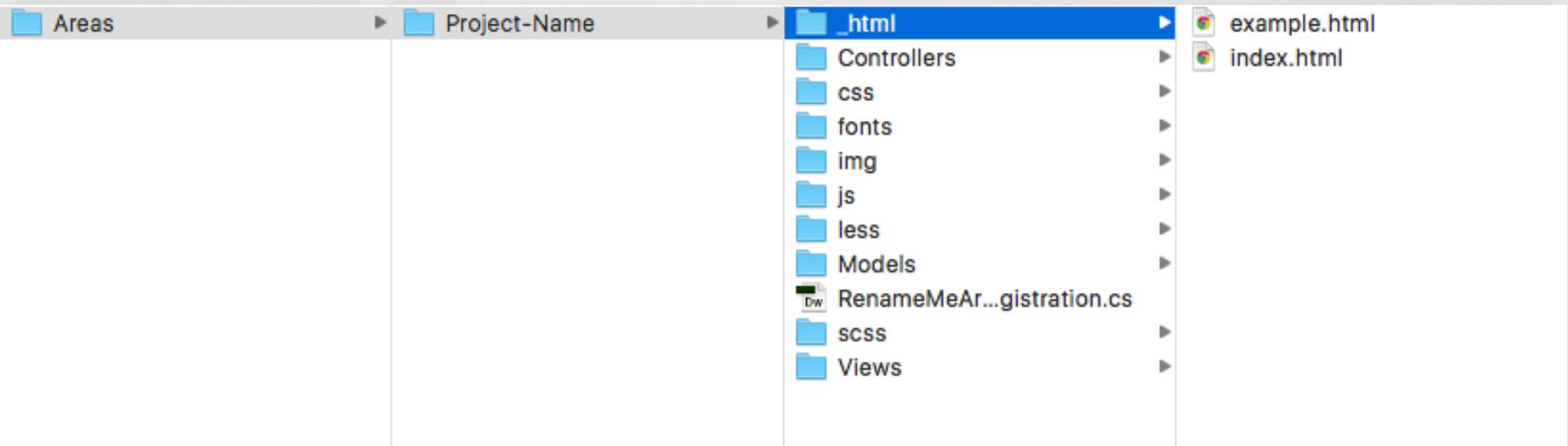
- Renamed to a specific project/site name



# File Structure

## HTML “Sandbox” folder

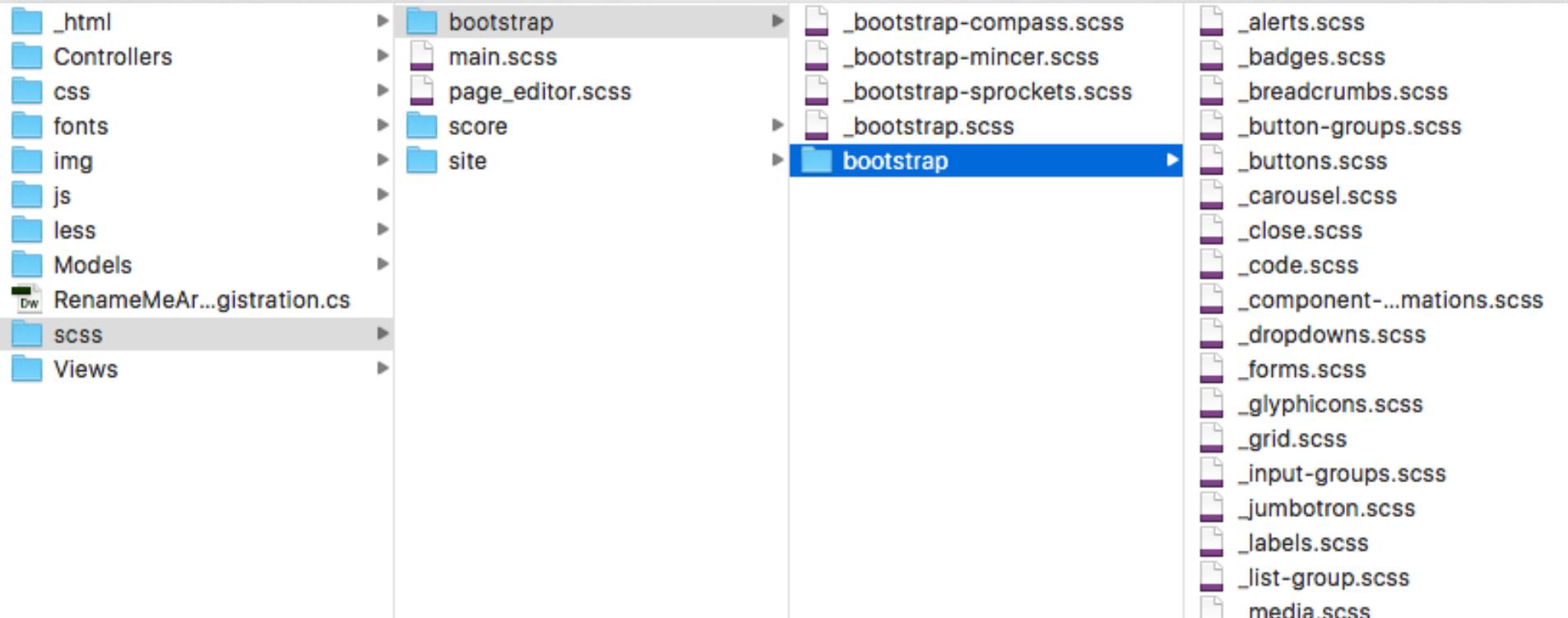
- Local HTML that is for your use - *Does not get deployed*
- example.html provided for “kitchen sink” examples
- index.html “boilerplate” provided for start-up



# File Structure

## Bootstrap Folder

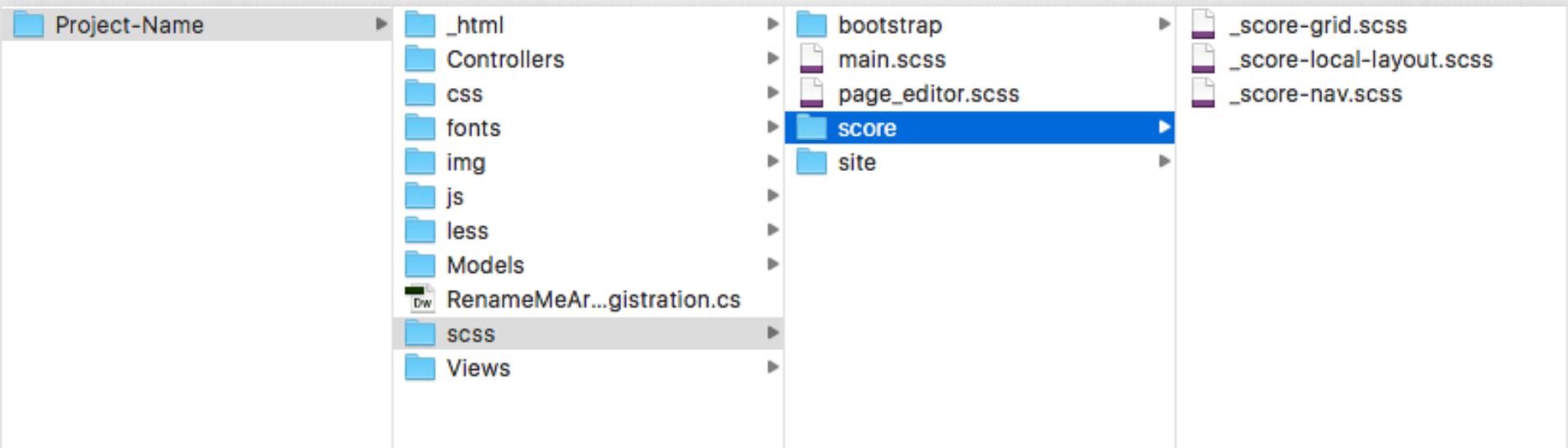
- Contains all of Bootstrap's default Sass files
- Location to access Bootstrap's variables and mixins (if needed)



# File Structure

## SCORE Folder

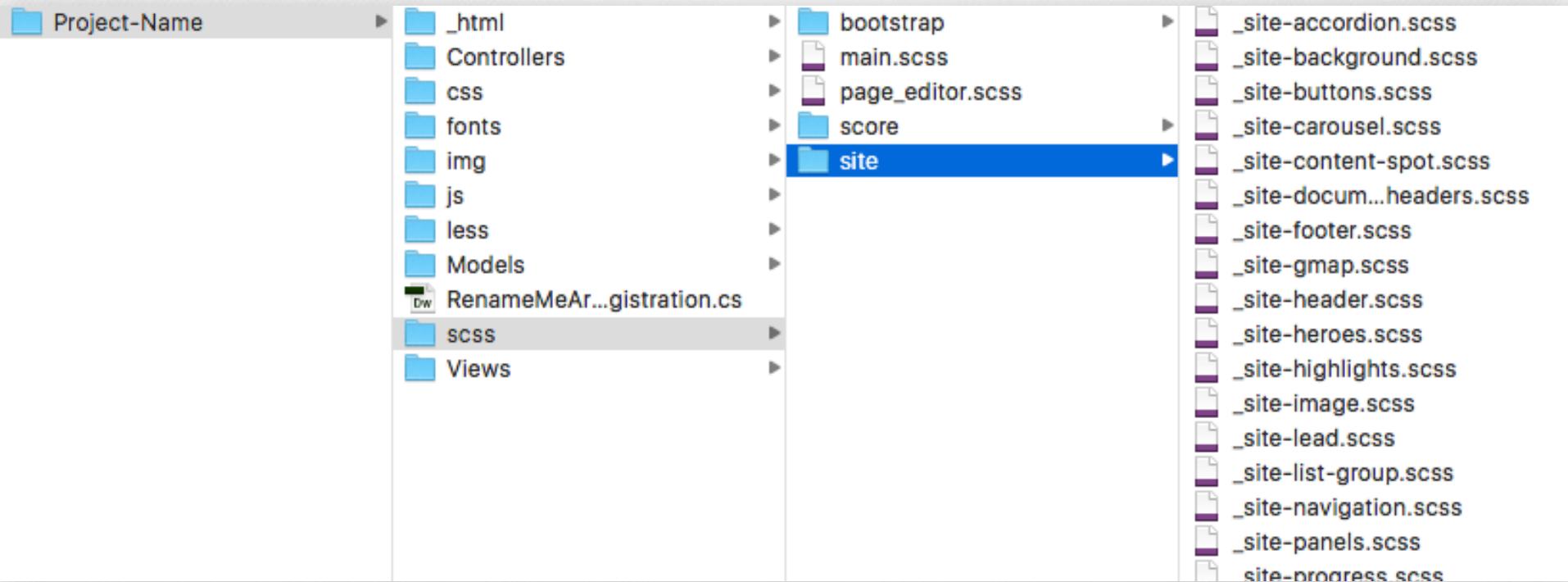
- Contains all of SCORE's default Sass files
- Location to modify SCORE-specific styles (if needed)



# File Structure

## Site Folder

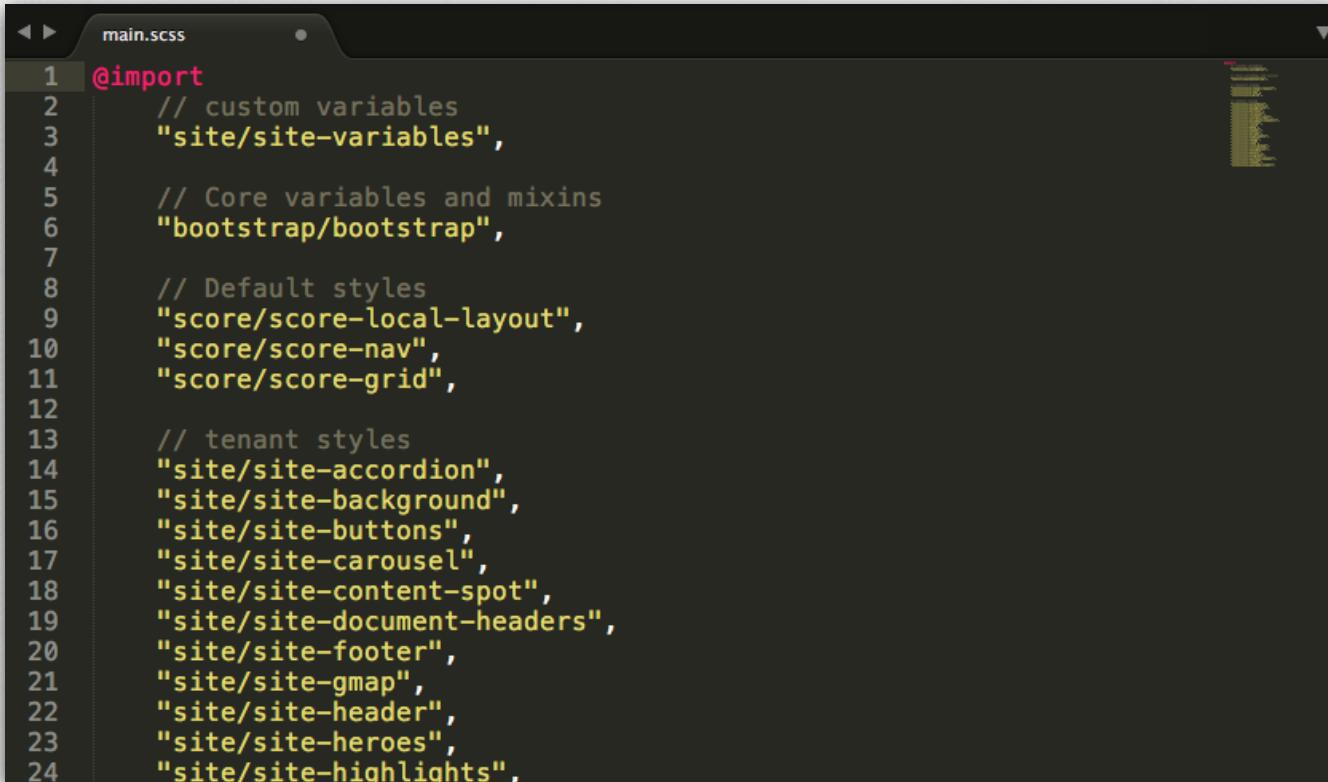
- Location to modify/include all site-specific styles
- Create site-specific variables and mixins



# File Structure

main.scss

- main.scss imports all Bootstrap, SCORE, and site Sass files
- Compiles to main.css



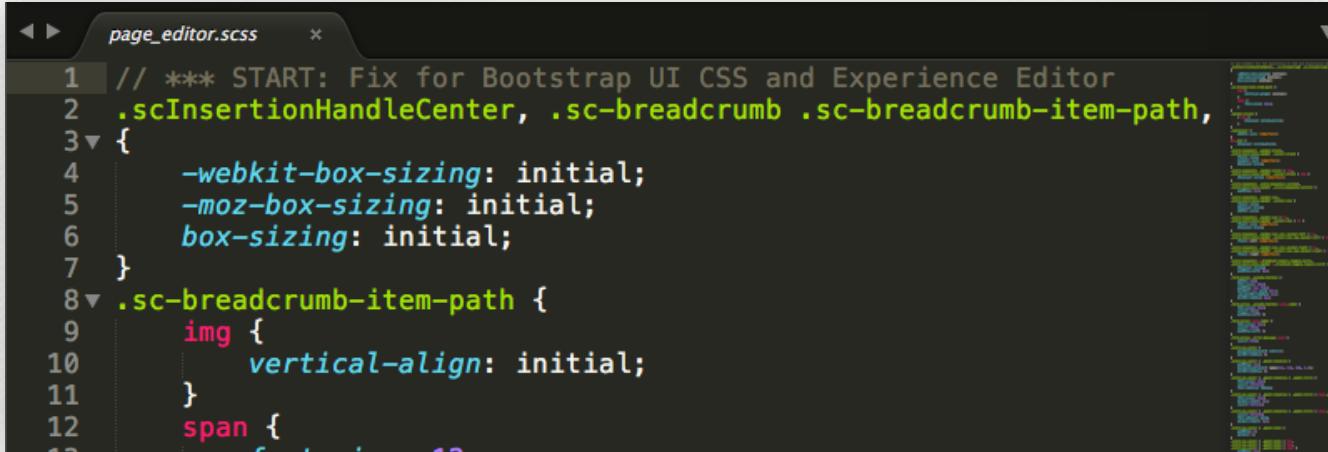
The screenshot shows a code editor window with the file 'main.scss' open. The code is a list of @import statements, each preceded by a line number from 1 to 24. The imports are categorized into three groups: custom variables, core variables and mixins, and tenant styles. The code is written in SCSS (Sass) syntax.

```
1 @import
2   // custom variables
3   "site/site-variables",
4
5   // Core variables and mixins
6   "bootstrap/bootstrap",
7
8   // Default styles
9   "score/score-local-layout",
10  "score/score-nav",
11  "score/score-grid",
12
13  // tenant styles
14  "site/site-accordion",
15  "site/site-background",
16  "site/site-buttons",
17  "site/site-carousel",
18  "site/site-content-spot",
19  "site/site-document-headers",
20  "site/site-footer",
21  "site/site-gmap",
22  "site/site-header",
23  "site/site-heroes",
24  "site/site-highlights".
```

# File Structure

page\_editor.scss

- Styles are specific to the SCORE experience editor
- Components may need to render differently in experience editor via CSS
- Components initially inherit main.css
- page\_editor.css *overrides* main.css and can include additional styles if needed



The screenshot shows a code editor window with the file "page\_editor.scss" open. The code is written in SCSS (Sass) and contains CSS declarations. The code is as follows:

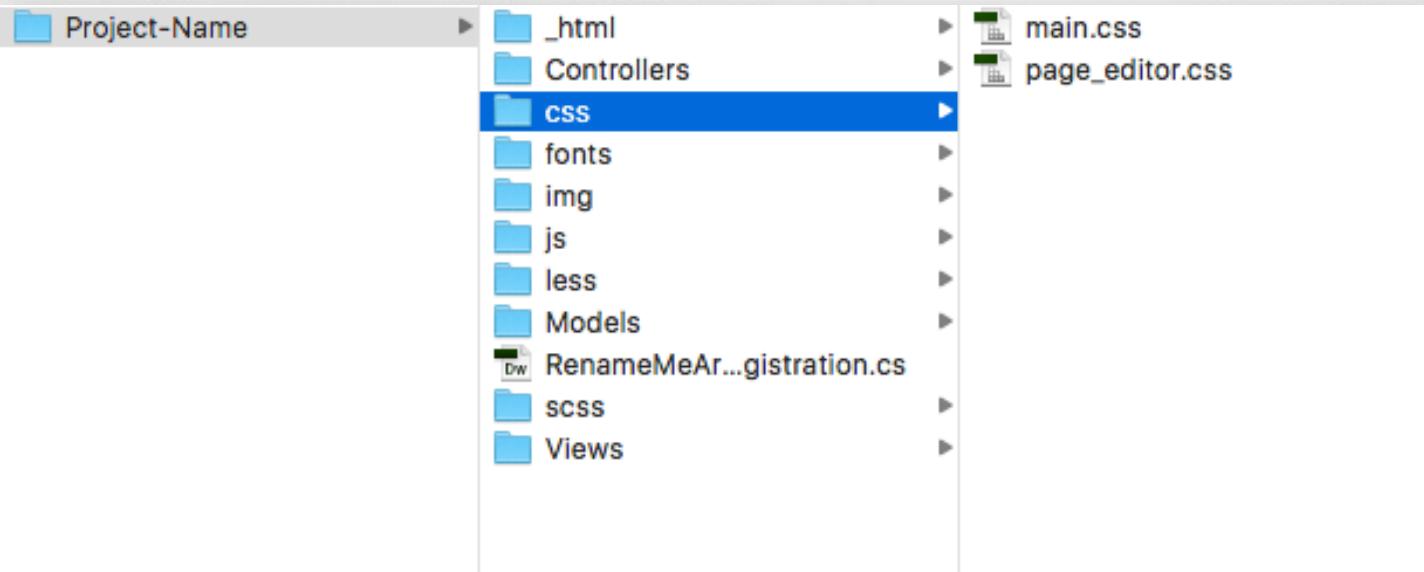
```
// *** START: Fix for Bootstrap UI CSS and Experience Editor
.scInsertionHandleCenter, .sc-breadcrumb .sc-breadcrumb-item-path,
{
    -webkit-box-sizing: initial;
    -moz-box-sizing: initial;
    box-sizing: initial;
}
.sc-breadcrumb-item-path {
    img {
        vertical-align: initial;
    }
    span {
        font-size: 1em;
    }
}
```

The code editor has a dark theme with syntax highlighting. The sidebar on the right shows a list of other files in the project.

# File Structure

## Project CSS files

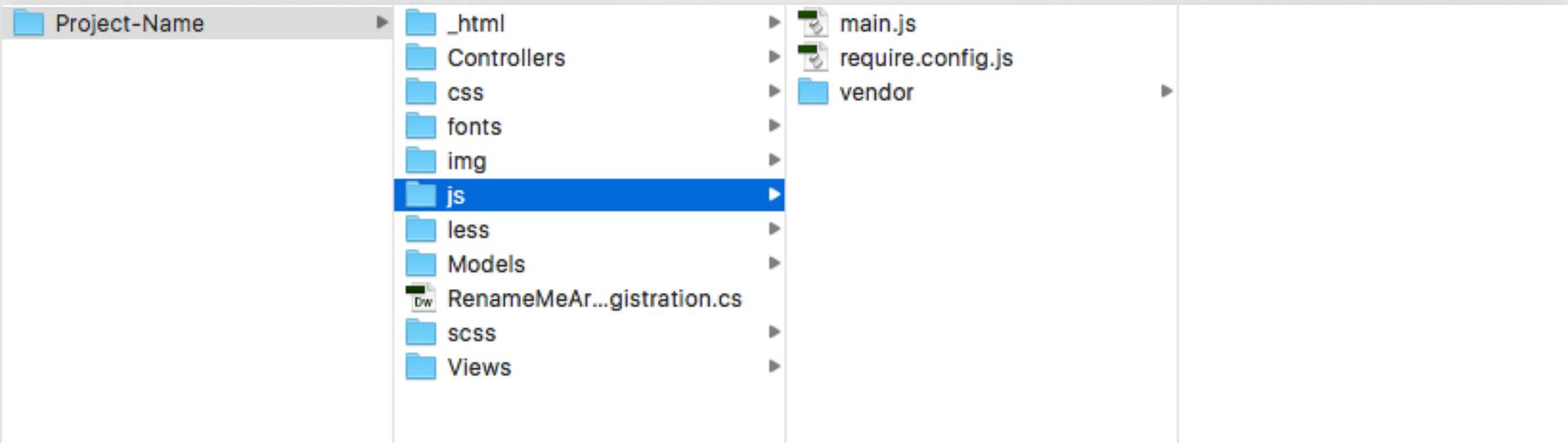
- **NEVER** edit CSS styles directly
- CSS files are **ONLY** to be compiled via Sass



# File Structure

## Project JS files

- SCORE 2.0 now uses [require.js](#) to deliver JavaScript files to your project
- Works in conjunction with main.js



# 04

## Where To Start?

Approaches to project development



# Where to start?

Two approaches to begin project development

- Style first, build last (you write HTML)  
★ We recommend this approach for any custom components only
- Build first, style last (you are given HTML)  
★ We recommend this approach for all SCORE components



# Where to start?

Build full composition pages **or** standalone components

- All of which are created/stored in \_html “Sandbox” folder for use
- Build/save individual HTML pages as POCs and keep them maintained
- Build/save an HTML page with all components and keep it maintained

05

Bootstrap

HTML/CSS/JS

# Bootstrap

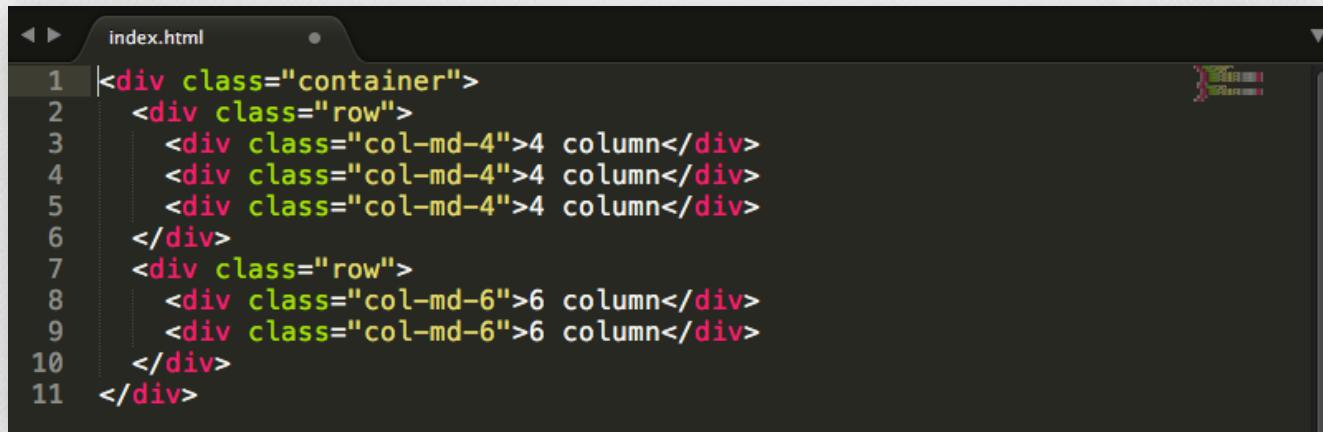
## What is Bootstrap?

- Naturally responsive UI framework
- Based on Sass pre-processor
- Provides a native responsive grid
- Source code includes:
  - Styling for native grid
  - Styling for over 20 native components
  - Javascript file for JS dependent components
- Reference: [Bootstrap 3](#)

# Bootstrap

## Using the grid

- Every page uses a container that holds rows and columns
- Bootstrap uses a 12-column layout per row (by default)
- Each element uses a specific class to pull in native Bootstrap styling
  - Example class names: `.container`, `.row`, `.col-md-4`



A screenshot of a code editor window titled "index.html". The code is written in HTML and shows a basic grid structure:

```
1 <div class="container">
2   <div class="row">
3     <div class="col-md-4">4 column</div>
4     <div class="col-md-4">4 column</div>
5     <div class="col-md-4">4 column</div>
6   </div>
7   <div class="row">
8     <div class="col-md-6">6 column</div>
9     <div class="col-md-6">6 column</div>
10  </div>
11 </div>
```

- Reference: [How Bootstrap's grid works](#)

# Bootstrap

## The Anatomy of Bootstrap's Grid

- “.container”
  - width based on screen size
  - 15px padding on left and right
  - margin: 0 auto; (always horizontally centered)

div.container 1170px x 288px

Heading	Heading	Heading
Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui. <a href="#">View details »</a>	Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui. <a href="#">View details »</a>	Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. <a href="#">View details »</a>

# Bootstrap

## The Anatomy of Bootstrap's Grid

- “.row”
  - width is 100% of parent
  - -15px margin on left and right
  - Parent should either be a container or a column

**Heading**

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

**Heading**

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

**Heading**

Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.

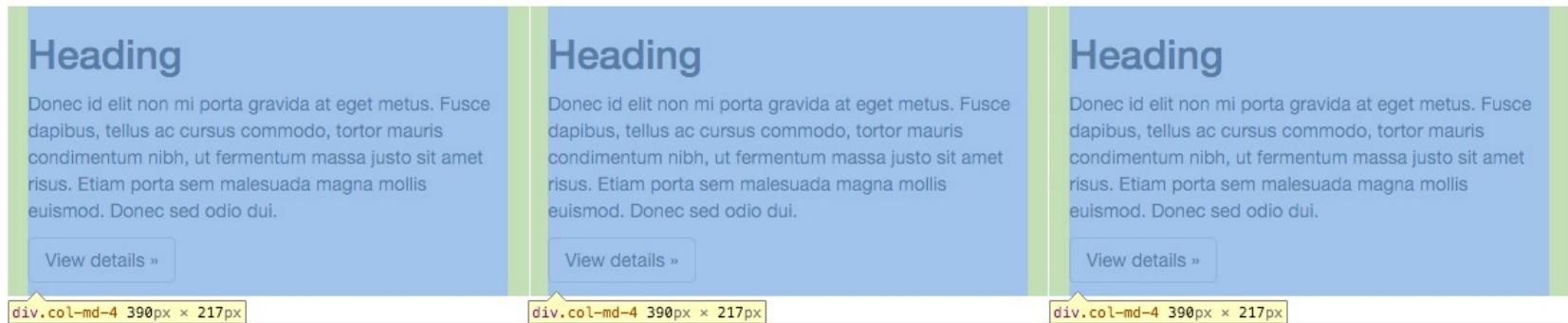
[View details »](#)

div.row 1170px x 217px

# Bootstrap

## The Anatomy of Bootstrap's Grid

- “.col-” (Column)
  - width is specified by developer using a class or mixin (Bootstrap uses 12-column grid)
  - 15px padding on left and right (30px “gutter” between columns)
  - ALWAYS inserted into a row



# Bootstrap

## The Anatomy of Bootstrap's Grid

- Nested row with columns:
  - To break up a column into more columns, first insert a nested row

### Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

### Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

### Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

### Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

`div.row 780px x 217px`

# Bootstrap

## The Anatomy of Bootstrap's Grid

- Nested row with columns:
  - Insert columns into the row like normal
  - `.col-lg-6` will be half the width of the parent `.row`

### Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

### Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

### Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

`div.col-md-6 390px × 217px`

### Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

`div.col-md-6 390px × 217px`

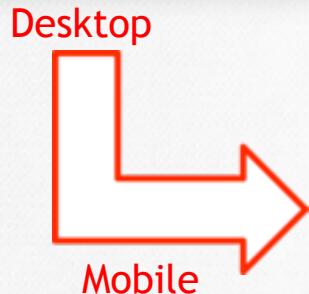
# Bootstrap

## Mobile Breakpoints

- Pixel, em, or rem value representing a screen width
  - Bootstrap uses pixels, defines 4 sizes: xs, sm, md, lg
- Mobile sizes do not have enough horizontal real estate for most columns to appear next to each other
- Grid columns “wrap” when screen width is less than mobile breakpoint size
  - All columns become 100% of screen width
  - Leftmost column becomes top column, columns to the right stack below in order

## Heading 1

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

## Heading 2

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

## Heading 3

Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.

[View details »](#)

## Heading 1

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

## Heading 2

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

## Heading 3

Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.

[View details »](#)

# Bootstrap

## Column Operations

- Offset
  - Moves columns to the right a specified number of column units
  - Uses class .col-md-offset-\*
    - Example: col-md-offset-4 increases left margin by 4 column units



The screenshot shows a code editor window titled "index.html". The code is as follows:

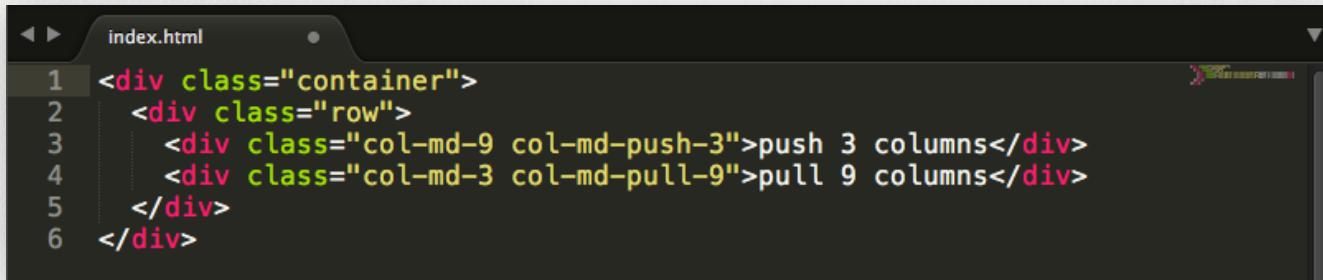
```
1 <div class="container">
2   <div class="row">
3     <div class="col-md-9 col-md-offset-3">offset 3 columns</div>
4   </div>
5 </div>
```

The code uses Bootstrap's grid system classes to create a row with a total of 12 columns. A column with the class "col-md-9 col-md-offset-3" is defined, which means it occupies 9 columns and has a left margin of 3 columns, effectively offsetting the first 3 columns.

# Bootstrap

## Column Operations

- Order
  - Switches order of columns
  - Uses classes `.col-md-push-*` and `.col-md-pull-*`



The screenshot shows a code editor window titled "index.html". The code is as follows:

```
1 <div class="container">
2   <div class="row">
3     <div class="col-md-9 col-md-push-3">push 3 columns</div>
4     <div class="col-md-3 col-md-pull-9">pull 9 columns</div>
5   </div>
6 </div>
```

The code demonstrates the use of Bootstrap's column ordering classes `.col-md-push-3` and `.col-md-pull-9` within a `row` and `container`.

# Bootstrap

## Column Operations

- **Responsive utilities**
  - Show or hide columns based on viewport width
  - Uses classes `.visible-**` or `.hidden-**`
    - Example: `.hidden-md` is hidden on screen sizes 992px - 1200px wide

	Extra small devices Phones (<768px)	Small devices Tablets ( $\geq 768px$ )	Medium devices Desktops ( $\geq 992px$ )	Large devices Desktops ( $\geq 1200px$ )
<code>.visible-xs-*</code>	Visible	Hidden	Hidden	Hidden
<code>.visible-sm-*</code>	Hidden	Visible	Hidden	Hidden
<code>.visible-md-*</code>	Hidden	Hidden	Visible	Hidden
<code>.visible-lg-*</code>	Hidden	Hidden	Hidden	Visible
<code>.hidden-xs</code>	Hidden	Visible	Visible	Visible
<code>.hidden-sm</code>	Visible	Hidden	Visible	Visible
<code>.hidden-md</code>	Visible	Visible	Hidden	Visible
<code>.hidden-lg</code>	Visible	Visible	Visible	Hidden

# Bootstrap

## Native Components

- Bootstrap styles native components using specific class names:
  - .nav, .navbar, .carousel, .jumbotron, .nav-tabs, .collapse, etc.
  - Note: by default, SCORE components inherit Bootstrap's native component styles (via LESS/Sass as of 2.0)
- bootstrap.js included in project scaffolding for use with components that require JavaScript
  - Carousel, tab sets, accordions, navigation
- References: [CSS components](#) / [JS components](#)



06

## SCORE HTML

Default component markup

# SCORE HTML

## The Basics

- Markup is intended to be portable
  - No boundaries for the grid
  - Markup is lightweight
  - Unbound to any specific framework (w/ an exception for JS components)
- All SCORE components are simply prefixed with “score-”
  - Examples:
    - <div class=“score-page”>
    - <div class=“score-structural”>
    - <div class=“score-highlight”>
- All SCORE components have the ability for *class selections*
  - Examples:
    - <div class=“score-page custom-name-1”>
    - <div class=“score-page custom-name-2”>
    - <div class=“score-highlight custom-highlight-class”>

# SCORE HTML



## Review Confluence Documentation

- [SCORE 2.0 Documentation](#)
  - Container
  - Content
  - Navigation
- Includes example screenshot and HTML output
- Older versions are child pages of current version

# SCORE HTML



## HTML “Sandbox” Folder

- **example.html**
  - “Kitchen sink” provided to help you tie-in SCORE component documentation and reveal how they are typically used
  - Provides a great reference point for components in your projects
- **index.html**
  - Boilerplate to help you build a local HTML page with links to CSS and JS
- **Sandbox is your playground**
  - Create new HTML pages here for maintenance of your SCORE and/or custom components
  - This folder is specifically for you as a front-end dev
  - Back-end dev will reference any custom component you have created here

# Styling & Inheritance with Sass

07



7.1

# Styling with Sass

Writing in scss and how it works for you

# Sass

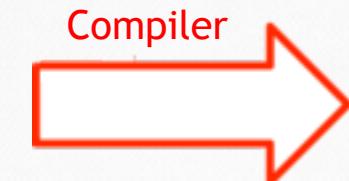
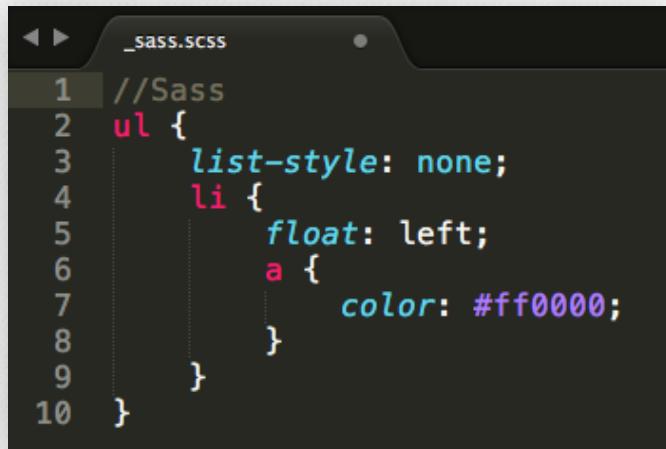


## Sass Basics

- CSS pre-processor (compiles into CSS)
  - All valid CSS is valid Sass
- Makes writing CSS easier with many added features
- Allows nested styles, which is *amazing* - but can *easily* get out of control
  - Try to keep nesting limited to 4 levels deep
- References: [Sass Tutorial](#), [Sass Style Guide](#)

# Sass

## Sass Compiling

The Sass logo is a stylized, purple, handwritten-style word "Sass".The CSS logo is a blue shield-shaped icon containing a white stylized letter "C".  
**css**

```
_sass.scss
1 //Sass
2 ul {
3     list-style: none;
4     li {
5         float: left;
6         a {
7             color: #ff0000;
8         }
9     }
10 }
```

A screenshot of a code editor showing a Sass file named "\_sass.scss". The code defines a class "Sass" which contains a rule for "ul" elements. Inside the "ul" rule, there is a nested "li" rule and an even more nested "a" rule with a color of "#ff0000". Line numbers 1 through 10 are visible on the left.

```
_sass.css
1 ul {
2     list-style: none;
3 }
4 ul li {
5     float: left;
6 }
7 ul li a {
8     color: #ff0000;
9 }
```

A screenshot of a code editor showing the resulting CSS file named "\_sass.css". The code is identical to the Sass file above, but it has been compiled into standard CSS syntax. The class "Sass" has been removed, and the rules are now defined directly under their respective selector levels.

# Sass



Command Line

Mac

Windows



Ruby



Prepros

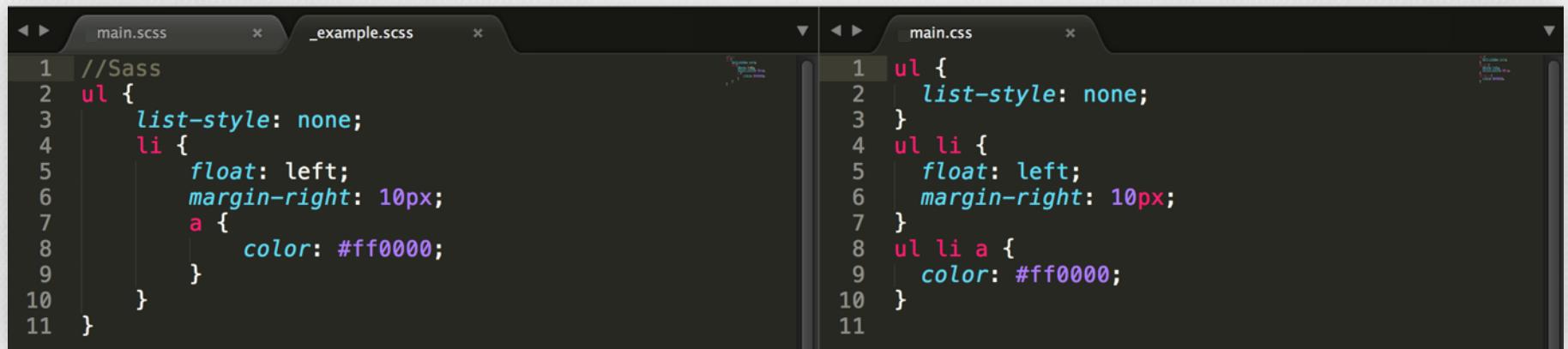
- Reference: [Installing a compiler](#)

# Exercise: Installing a Compiler

1. Navigate to <https://prepros.io/>
2. Download compiler
3. Open the SCORE-2.0-UI-class folder
4. Find the “RenameMeArea” folder and rename it as your “(First Name)-Project”
5. Add the project to your compiler
6. Navigate to the scss folder
7. Manually compile main.scss

# Sass

## Good Nesting Example



The screenshot shows a code editor with two tabs: `main.scss` and `_example.scss` on the left, and `main.css` on the right. The `main.scss` tab contains the following Sass code:

```
1 //Sass
2 ul {
3   list-style: none;
4   li {
5     float: left;
6     margin-right: 10px;
7     a {
8       color: #ff0000;
9     }
10 }
11 }
```

The `_example.scss` tab contains the following Sass code:

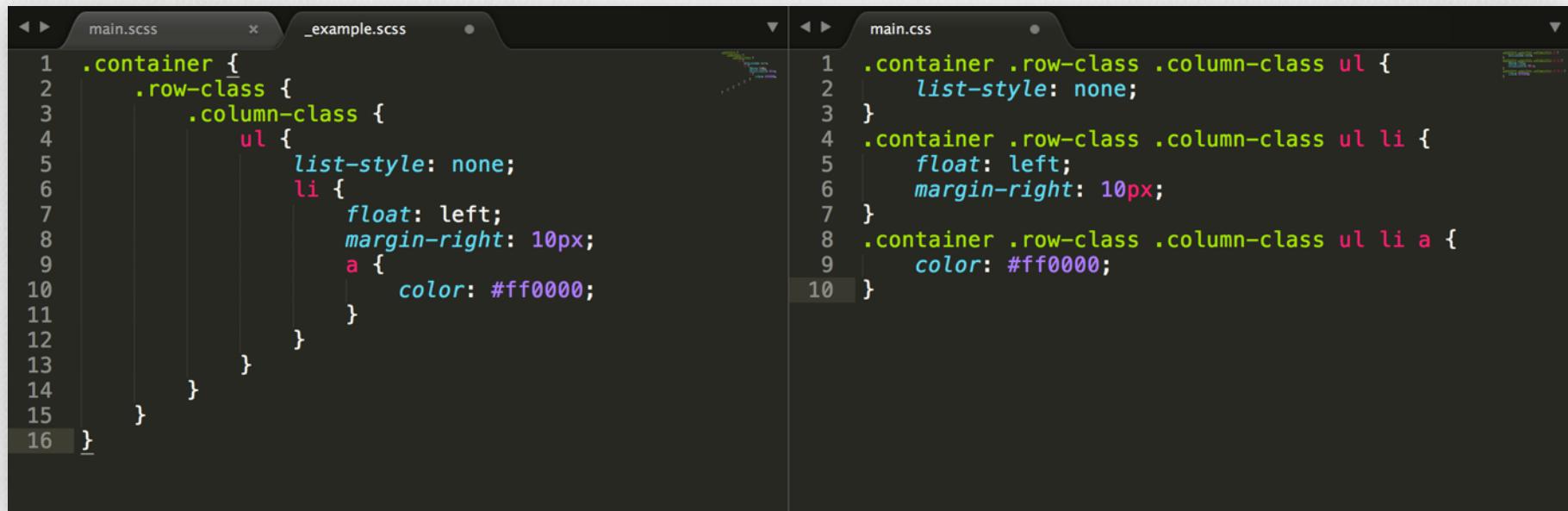
```
1 ul {
2   list-style: none;
3 }
4 ul li {
5   float: left;
6   margin-right: 10px;
7 }
8 ul li a {
9   color: #ff0000;
10 }
11 }
```

The `main.css` tab on the right shows the generated CSS output:

```
1 ul {
2   list-style: none;
3 }
4 ul li {
5   float: left;
6   margin-right: 10px;
7 }
8 ul li a {
9   color: #ff0000;
10 }
11 }
```

# Sass

## Bad Nesting Example



```
main.scss
1 .container {
2   .row-class {
3     .column-class {
4       ul {
5         list-style: none;
6         li {
7           float: left;
8           margin-right: 10px;
9           a {
10             color: #ff0000;
11           }
12         }
13       }
14     }
15   }
16 }
```

```
main.css
1 .container .row-class .column-class ul {
2   list-style: none;
3 }
4 .container .row-class .column-class ul li {
5   float: left;
6   margin-right: 10px;
7 }
8 .container .row-class .column-class ul li a {
9   color: #ff0000;
10 }
```

# Sass

## Nesting Media Queries

- Media queries can be nested to specify mobile behavior for individual elements

The image shows a code editor interface with two tabs: 'main.scss' on the left and 'main.css' on the right. The 'main.scss' tab contains the following Sass code:

```
1 //Sass
2 .class-1 {
3     @media only screen and (min-width: 768px) {
4         color: #ff0000;
5     }
6     @media only screen and (min-width: 992px) {
7         color: #2b2b2b;
8     }
9 }
```

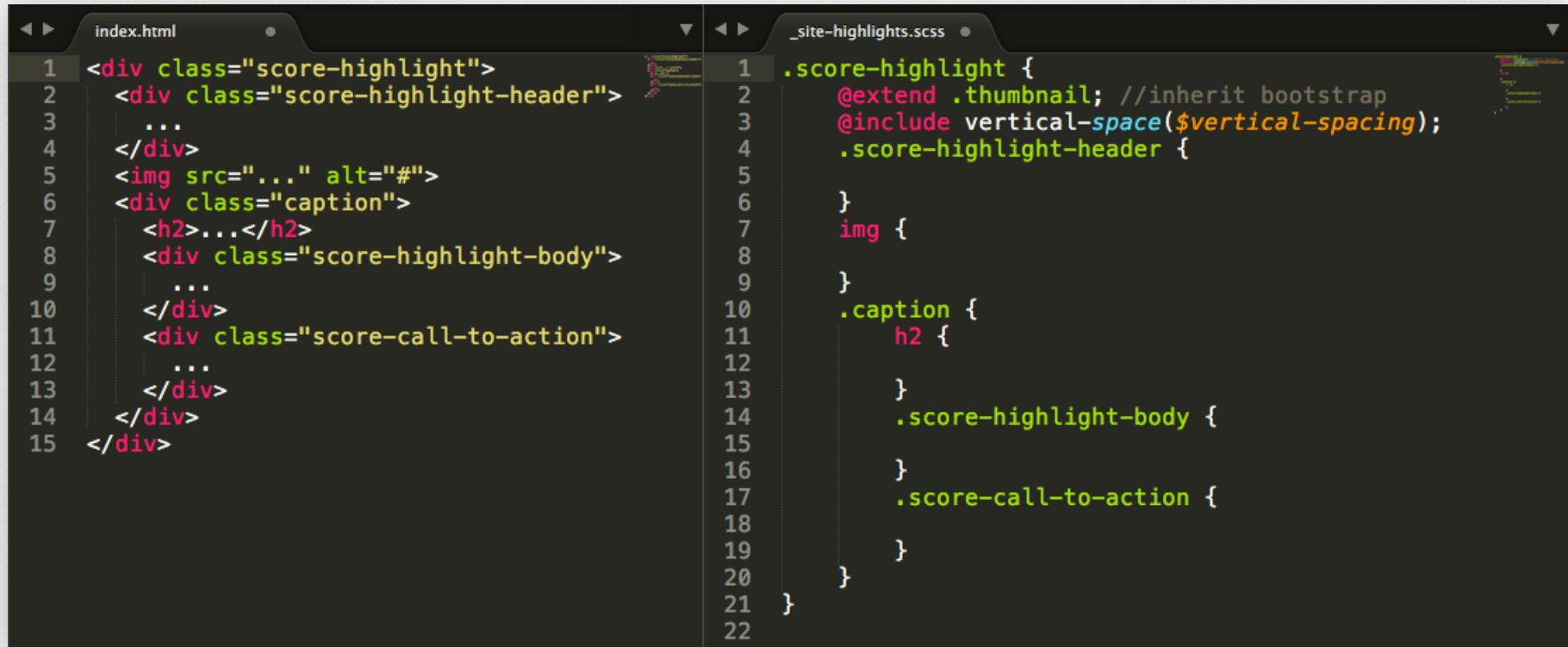
The 'main.css' tab contains the generated CSS code:

```
1 @media only screen and (min-width: 768px) {
2     .class-1 {
3         color: #ff0000;
4     }
5 }
6 @media only screen and (min-width: 992px) {
7     .class-1 {
8         color: #2b2b2b;
9     }
10 }
11 }
```

# Sass

## Tying Up The Markup & Styling

- Default styles are pre-scaffolded to reflect default markup
- Empty nests do not compile unnecessary CSS!



The image shows a code editor with two tabs: "index.html" and "\_site-highlights.scss".

**index.html:**

```
1 <div class="score-highlight">
2   <div class="score-highlight-header">
3     ...
4   </div>
5   
6   <div class="caption">
7     <h2>...</h2>
8     <div class="score-highlight-body">
9       ...
10    </div>
11    <div class="score-call-to-action">
12      ...
13    </div>
14  </div>
15 </div>
```

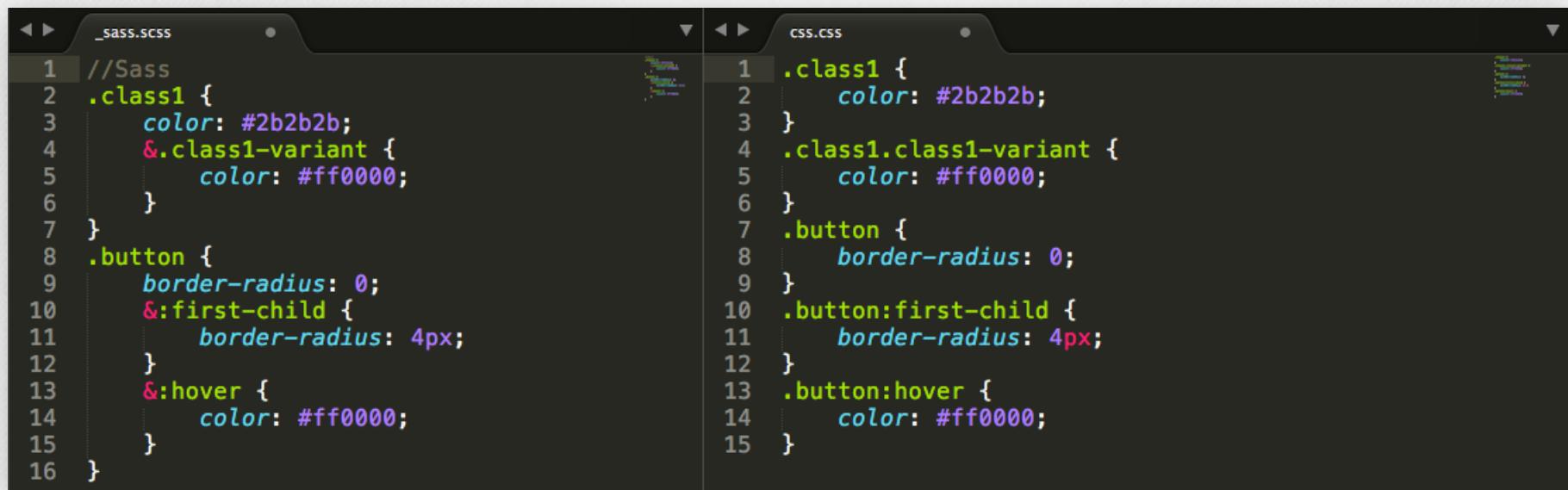
**\_site-highlights.scss:**

```
1 .score-highlight {
2   @extend .thumbnail; //inherit bootstrap
3   @include vertical-space($vertical-spacing);
4   .score-highlight-header {
5   }
6   img {
7   }
8   .caption {
9     h2 {
10   }
11   .score-highlight-body {
12   }
13   .score-call-to-action {
14   }
15 }
16 }
17 }
18 }
19 }
20 }
21 }
22 }
```

# Sass

## Sass Operators

- &
  - Appends statement to immediate parent
  - Commonly used for compound classes and pseudo elements



The image shows a code editor with two panes. The left pane is titled '\_sass.scss' and contains the following Sass code:

```
//Sass
.class1 {
  color: #2b2b2b;
  &.class1-variant {
    color: #ff0000;
  }
}
.button {
  border-radius: 0;
  &:first-child {
    border-radius: 4px;
  }
  &:hover {
    color: #ff0000;
  }
}
```

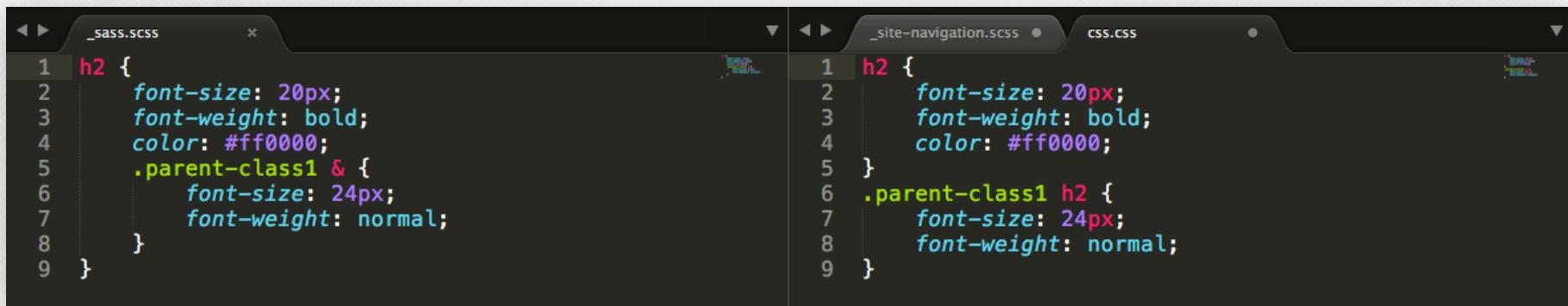
The right pane is titled 'css.css' and contains the generated CSS code:

```
.class1 {
  color: #2b2b2b;
}
.class1.class1-variant {
  color: #ff0000;
}
.button {
  border-radius: 0;
}
.button:first-child {
  border-radius: 4px;
}
.button:hover {
  color: #ff0000;
}
```

# Sass

## Sass Operators

- &
  - Can also go at the end of a selector to append the PARENT statement to the end



The image shows a code editor with two panes. The left pane is titled '\_SASS.scss' and contains the following Sass code:

```
1 h2 {  
2   font-size: 20px;  
3   font-weight: bold;  
4   color: #ff0000;  
5   .parent-class1 & {  
6     font-size: 24px;  
7     font-weight: normal;  
8   }  
9 }
```

The right pane is titled '\_site-navigation.scss' and contains the generated CSS code:

```
1 h2 {  
2   font-size: 20px;  
3   font-weight: bold;  
4   color: #ff0000;  
5 }  
6 .parent-class1 h2 {  
7   font-size: 24px;  
8   font-weight: normal;  
9 }
```

# Sass

## Sass Operators

- >
  - Direct child selector (same as in CSS)
  - Can be nested and combined with “&”

The image shows a code editor interface with two tabs: '\_sass.scss' on the left and 'CSS.css' on the right. The '\_sass.scss' tab contains the following Sass code:

```
//Sass
.class1 {
  & > .class2 {
    color: #ff0000;
  }
}
```

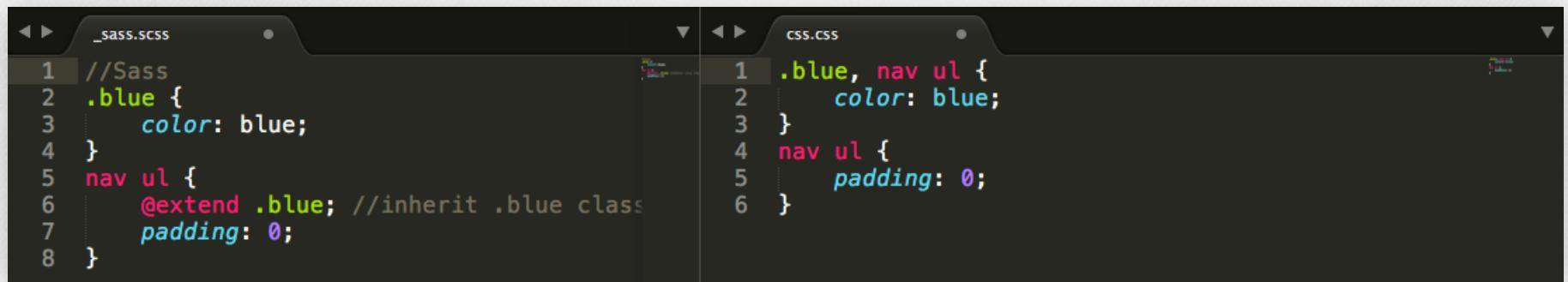
The 'CSS.css' tab contains the generated CSS code:

```
.class1 > .class2 {
  color: #ff0000;
}
```

# Sass

## Extending Classes

- @extend adds the current selector to the selector that is being extended
- Reference: [The extend concept](#)



The screenshot shows a code editor with two tabs: '\_sass.scss' and 'CSS.CSS'. The '\_sass.scss' tab contains the following Sass code:

```
//Sass
.blue {
  color: blue;
}
nav ul {
  @extend .blue; //inherit .blue class
  padding: 0;
```

The 'CSS.CSS' tab contains the generated CSS code:

```
.blue, nav ul {
  color: blue;
}
nav ul {
  padding: 0;
```

# Inheriting Bootstrap via Sass

How SCORE 2.0 inherits default styles

# Sass & Inheritance

## Inheritance

- Declared with @extend .class-name-you're-extending
- This is the magic that includes your framework's classes into your default component HTML
- Simply remove the extension if you need to remove default styles

```
_main.scss          _site-highlights.scss          _main.css
1  .score-highlight {           1  .score-highlight {           5910
2    @extend .thumbnail; //inherit bootstrap           5911   .thumbnail, .score-highlight {           5912     display: block;           5913     padding: 4px;           5914     margin-bottom: 20px;           5915     line-height: 1.42857;           5916     background-color: white;           5917     border: 1px solid #dddddd;           5918     border-radius: 4px;           5919     -webkit-transition: border 0.2s ease-in-           5920     -o-transition: border 0.2s ease-in-out;           5921     transition: border 0.2s ease-in-out;           5922   }
5923   .thumbnail > img, .score-highlight > img,           5924   .thumbnail a > img,           5925   .score-highlight a > img {           5926     display: block;           5927   }
```

# Sass & Inheritance

Markup & Styling: *Then...* (version 1.5 and below)

- Default styles were inherited via HTML and essentially “baked-in”
- Default styles had to be overridden in stylesheets
- Time consuming overriding Bootstrap styles (borders, radius, background-color, etc.)

The screenshot shows a code editor with two tabs: 'index.html' and 'site-highlight.less'. The 'index.html' tab contains the following code:

```
1 <div class="thumbnail score-component score-  
content score-image score-highlight">  
2 ...  
3 </div>
```

The 'score-highlight' class is highlighted in yellow. The 'site-highlight.less' tab contains the following code:

```
1 .score-highlight {  
2     //overriding Bootstrap styles went here  
3     ...  
4     ...  
5     ...  
6     ...  
7 }
```

# Sass & Inheritance

Markup & Styling: *Now!* (version 2.0)

- Default style class names are removed from the HTML
- Default styles are simply inherited via LESS/Sass with @extend
- Removing necessary default styles is easy

The screenshot shows a code editor with two tabs: 'index.html' and '\_site-highlights.scss'. The 'index.html' tab contains the following code:

```
1 <div class="score-highlight">
2 ...
3 </div>
```

The 'score-highlight' class is highlighted in yellow. The '\_site-highlights.scss' tab contains the following SCSS code:

```
1 .score-highlight {
2   ...
3 }
4
```

The class '.score-highlight' is followed by an '@extend' directive that inherits styles from '.thumbnail'.

# Sass

## Sass Variables

- Declared locally or globally and prefixed with \$
- Can be a color, font, another variable, etc.
- Code is easily maintained - changing the value of the variable will change all occurrences
- Bootstrap includes native variables for colors, fonts, and common mobile breakpoints

The screenshot shows a code editor with three tabs: 'main.scss', '\_example.scss', and 'main.css'. The 'main.scss' and '\_example.scss' tabs are on the left, while the 'main.css' tab is on the right. The 'main.scss' file contains the following code:

```
1 //Sass variables
2 $brand-color: #ff0000;
3
4 //Sass styles
5 header, footer {
6   background-color: $brand-color;
7 }
8 h1 {
9   color: $brand-color;
10 }
```

The '\_example.scss' file contains the same code as 'main.scss'. The 'main.css' file on the right shows the generated CSS output:

```
1 header, footer {
2   background-color: red;
3 }
4 h1 {
5   color: red;
6 }
7
8
```

# LAB 2

1. Navigate to scss > site > \_site-variables.scss
2. Define a new variable, \$button-color
3. Navigate to \_site-buttons.scss in the same folder
4. Change the background color of all SCORE button components (class of .score-button) to \$button-color
5. After successful compilation, open \_html > index.html in a browser
6. Change \$button-color to another color value
7. Compile main.scss and view index.html in the browser

# Sass

## Sass Mixins

- Declared locally or globally and prefixed with @mixin
- Create easily reusable and maintainable blocks of code
- Commonly used for styles that require browser prefixes
- Can accept parameters to use different values in different locations

```
_site-variables.scss
1 //Global custom variables
2
3 //default margin on content components
4 $vertical-spacing: 30px;
5
6 //Global custom mixins
7 @mixin vertical-space($margin) {
8   margin-bottom: $margin;
9 }
10 @mixin rounded-corners($radius) {
11   -webkit-border-radius: $radius;
12   -moz-border-radius: $radius;
13   -o-border-radius: $radius;
14   border-radius: $radius;
15 }

_site-highlights.scss
1 .score-highlight {
2   @include vertical-space($vertical-spacing);
3   @include rounded-corners(3px);
4 }
```

# Sass

## Sass Mixins

- Example of pixel to rem converter mixin

The screenshot shows a code editor with two tabs: '\_SASS.scss' on the left and 'CSS.CSS' on the right. The '\_SASS.scss' tab contains the following Sass code:

```
1 @mixin pxConvert($px) {  
2     font-size: ($px / 16) + rem;  
3 }  
4  
5 h2 {  
6     @include pxConvert(12);  
7     font-weight: bold;  
8     color: #ff0000;  
9 }  
10  
11
```

The 'CSS.CSS' tab contains the generated CSS output:

```
1 h2 {  
2     font-size: 0.75rem;  
3     font-weight: bold;  
4     color: #ff0000;  
5 }
```

# Sass

## Vertical Spacing

- Set to 30px for vertical space by default

```
_site-variables.scss
1 //Global custom variables
2
3 //default margin on content components
4 $vertical-spacing: 30px;
5
6 //Global custom mixins
7 @mixin vertical-space($margin) {
8   margin-bottom: $margin;
9 }
```

```
_site-highlights.scss
1 .score-highlight {
2   @extend .thumbnail; //inherit bootstrap
3   @include vertical-space($vertical-spacing);
4   .score-highlight-header {
5
6   }
7   img {
8
9   }
10  .caption {
11    h2 {
12
13    }
14    .score-highlight-body {
15
16    }
17    .score-call-to-action {
18
19    }
20  }
21 }
22 }
```



# LAB 3

## Exercise: Using a Mixin

1. Navigate to scss > site > \_site-variables.scss
2. Define a new mixin called “scale-transition”, which will scale something using CSS3 animation
3. Navigate to \_site-buttons.scss
4. Insert scale-transition mixin into the button’s hover state
5. Compile main.scss and view index.html in the browser. Hover over a button to verify your update
6. Modify scale-transition mixin to accept a parameter for a scale size, use variable \$size
7. Insert 1.2 as a size value into the call to scale-transition in \_site-buttons.scss
8. Compile main.scss and hover over a button

# SCORE Components

08



8.1

# SCORE Containers

Understanding SCORE's grid system

# SCORE Containers

## Container Components

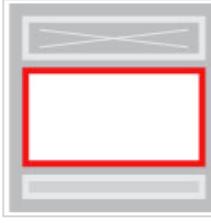
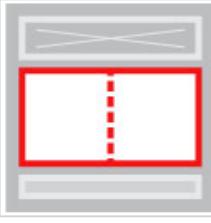
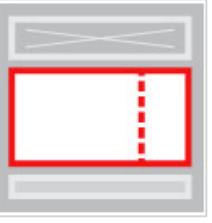
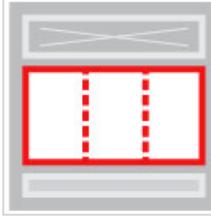
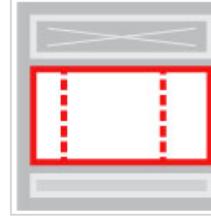
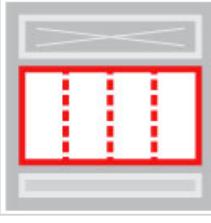
- Container components are simply components that hold other components (i.e. the grid)
- Container renderings:
  - Page Elements (Header, Footer)
  - Page Layout (.container + .row + .col-md-\*)
  - Inner Structure (.row + .col-md-\*)
  - Stripes
- Reference: [Container Documentation](#)

# SCORE Containers

## Page Layout

Select a Rendering  
Select the rendering that you want to use. Click Select to continue.

Open the Properties dialog box immediately.

CONTENT	DOCUMENT HEADERS	PAGE LAYOUT	STRIPES
	 WIDE SCREEN ONE COLUMN		
1 Column	1 Column Wide Screen	2 Column - Equal	2 Column - Large Left
			2 Column - Large Right
3 Column - Equal	3 Column - Large Middle	4 Column - Equal	

# SCORE Containers

## One-Column Page Layout Example:

- Bootstrap version on left - SCORE version on right
- Note the .container wrapper
- Page Layouts **CANNOT** be nested inside of each other

The image shows a code editor with two tabs open: "bootstrap.html" on the left and "score.html" on the right. Both files contain the same basic structure: a single-level row with a single column.

```
bootstrap.html
1 <!-- Bootstrap row with columns -->
2 <div class="container">
3   <div class="row">
4     <div class="col-md-12">
5       ...
6     </div>
7   </div>
8 </div>
```

```
score.html
1 <!-- SCORE one column -->
2 <div class="container">
3   <div class="score-column1">
4     <div class="score-center">
5       ...
6     </div>
7   </div>
8 </div>
```

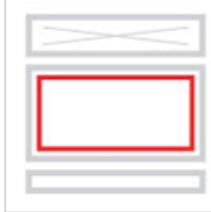
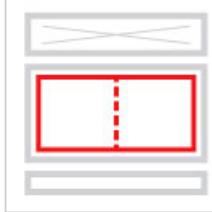
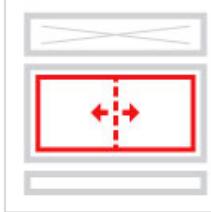
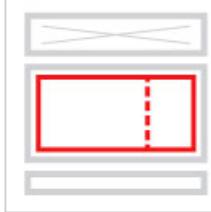
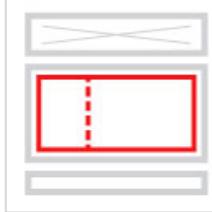
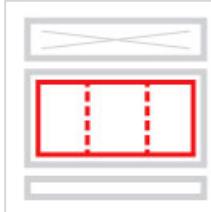
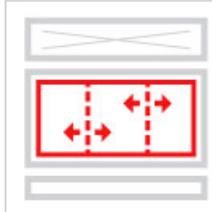
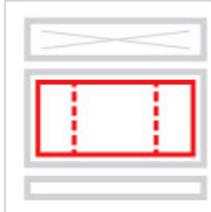
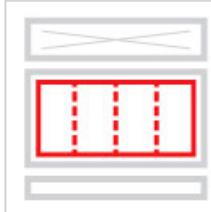
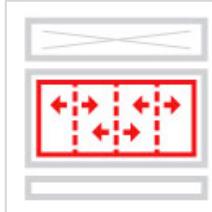
# SCORE Containers

## Inner Structure

Select a Rendering  
Select the rendering that you want to use. Click Select to continue.

Open the Properties dialog box immediately.

Buttons	Collections	Content	Document Headers	Features	Forms	INNER STRUCTURE
Navigation	Panels	Products	Search	Section Headers	Social	Stores

						
1 Column	2 Column - Equal	2 Column - Variable	2 Column - Wide Left	2 Column - Wide Right		
						
3 Column - Equal	3 Column - Variable	3 Column - Wide Middle	4 Column - Equal	4 Column - Variable		

# SCORE Containers

## Inner Structure Example:

- Bootstrap version on left - SCORE version on right
- Inner Structures must be nested inside of Page Layouts
- Inner Structures can be nested inside of each other

```
bootstrap.html
1  <!-- Bootstrap row with nested columns -->
2  <div class="container">
3    <div class="row">
4      <div class="col-md-12">
5        <div class="row">
6          <div class="col-md-6">
7            ...
8          </div>
9          <div class="col-md-6">
10         ...
11        </div>
12      </div>
13    </div>
14  </div>
15 </div>
```

```
score.html
1  <!-- SCORE row with nested columns -->
2  <div class="container">
3    <div class="score-column1">
4      <div class="score-center">
5        <div class="score-column2 equal">
6          <div class="score-left">
7            ...
8          </div>
9          <div class="score-right">
10         ...
11        </div>
12      </div>
13    </div>
14  </div>
15 </div>
```

# SCORE Containers

Stripe

Select a Rendering

Select the rendering that you want to use. Click Select to continue.

□ X

CONTENT DOCUMENT HEADERS PAGE LAYOUT STRIPES



Stripe

Open the Properties dialog box immediately.

Select Cancel

# SCORE Containers

## Stripe Example:

- SCORE stripe is a full-width wrapper
  - Used to separate SCORE page layouts for full-width backgrounds
- Background image/color and text color are applied as inline styles
- Page Layout is placed inside

```
bootstrap.html
```

```
1 <!-- Bootstrap full width wrapper -->
2 <div class="full-width-wrapper">
3   <div class="container">
4     <div class="row">
5       <div class="col-md-12">
6         ...
7       </div>
8     </div>
9   </div>
10 </div>
```

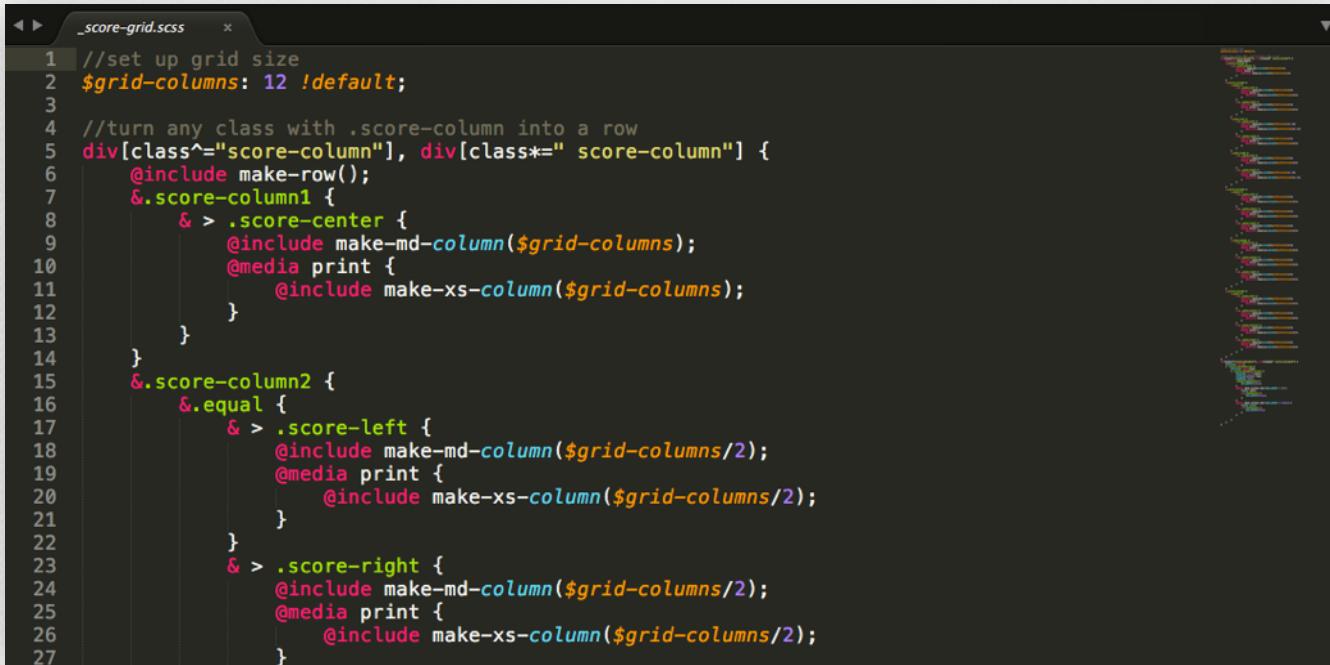
```
score.html
```

```
1 <!-- SCORE stripe -->
2 <div class="score-stripe">
3   <div class="container">
4     <div class="score-column1">
5       <div class="score-center">
6         ...
7       </div>
8     </div>
9   </div>
10 </div>
```

# SCORE Containers

## Responsive Grid Mixins

- Grid mixins make or modify rows and columns at the CSS level
- These mixins can be used to create custom layouts
  - Change default 2-column wide-left from an 8/4 pair into a 7/5 or 9/3 pair
  - Make default medium columns into large, small, or extra-small columns

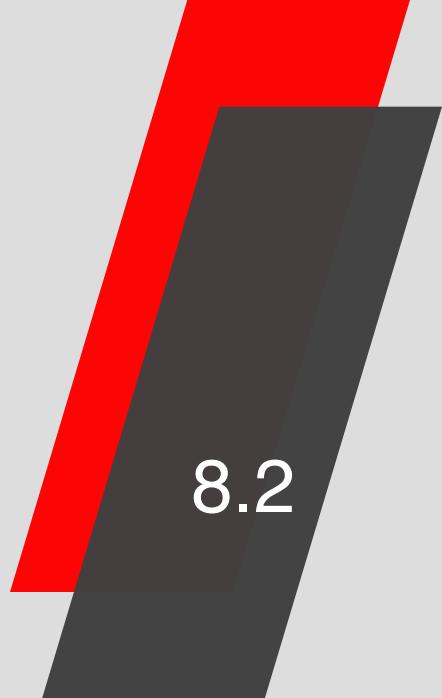


```
_score-grid.scss
1 //set up grid size
2 $grid-columns: 12 !default;
3
4 //turn any class with .score-column into a row
5 div[class^="score-column"], div[class*=" score-column"] {
6     @include make-row();
7     &.score-column1 {
8         & > .score-center {
9             @include make-md-column($grid-columns);
10            @media print {
11                @include make-xs-column($grid-columns);
12            }
13        }
14    }
15    &.score-column2 {
16        &.equal {
17            & > .score-left {
18                @include make-md-column($grid-columns/2);
19                @media print {
20                    @include make-xs-column($grid-columns/2);
21                }
22            }
23            & > .score-right {
24                @include make-md-column($grid-columns/2);
25                @media print {
26                    @include make-xs-column($grid-columns/2);
27                }
28        }
29    }
30}
```

# Exercise: Create Basic SCORE Grid Example

LAB  
4

1. Navigate to the “SCORE-2.0-UI-class” folder > SCORE\_grid\_example.pdf
2. Identify page layout
3. Identify inner structure layout
4. Identify stripes
5. Navigate to the “\_html” folder > index.html
6. Create page/structural markup
7. Use the SCORE Highlight component to place inside all of your columns



8.2

# SCORE Content

Understanding SCORE's content components

# SCORE Content

## Content Components

- Content components cover a large variety of elements to deliver content and drive user experience (mostly comprised of Bootstrap's components as default)
- Content Renderings:
  - Features
  - Buttons
  - Collections
  - Document Headers
  - Panels
  - Section Headers
- Reference: [Content Documentation](#)

# SCORE Content

REUSE!

- The key to gaining full advantage of SCORE is all about *reuse*.
  - Familiarize yourself with every SCORE component
  - Know when/how/where to use default components
  - Saves you time and money!
- Avoid creating custom components!
  - Custom components are usually inevitable (*but* you'll be surprised with what you can achieve with SCORE components)
  - Strive to make sure that custom components are your last resort
- Class Selections:
  - SCORE 2.0 now allows class selections for *ALL* components
  - Create many different versions of the same component
  - Available for page layouts and inner structures too
  - Extremely beneficial for UI devs AND content admins

# SCORE Content

## Highlight Component

Select a Rendering

Select the rendering that you want to use. Click Select to continue.

Open the Properties dialog box immediately.

BUTTONS	COLLECTIONS	CONTENT	DOCUMENT HEADERS	FEATURES	INNER STRUCTURE
NAVIGATION	PANELS	SECTION HEADERS	SOCIAL		

HighLight

# SCORE Content

## Highlight Example:

- Generated HTML on left - Stylesheet on right

The screenshot shows a code editor with two panes. The left pane displays the generated HTML code for a 'score-highlight' component, while the right pane displays the corresponding SCSS stylesheet.

**example.html**

```
1 <div class="score-highlight">
2   <div class="score-highlight-header"></div>
3   
4   <div class="caption">
5     <h2>...</h2>
6     <div class="score-highlight-body">
7       ...
8     </div>
9     <div class="score-call-to-action">
10    ...
11   </div>
12 </div>
13 </div>
```

**\_site-highlights.scss**

```
1 .score-highlight {
2   @extend .thumbnail; //inherit bootstrap
3   @include vertical-space($vertical-spacing);
4   .score-highlight-header {
5   }
6   img {
7   }
8   .caption {
9     h2 {
10   }
11   .score-highlight-body {
12   }
13   .score-call-to-action {
14   }
15   }
16   }
17   }
18   }
19   }
20   }
21   }
22 }
```

# SCORE Content

## Highlight with Class Selection Example:

- Generated HTML on left - Stylesheet on right

The screenshot shows a code editor with two files side-by-side. On the left is `example.html`, which contains the following HTML:

```
1 <div class="score-highlight all-new-class">
2   <div class="score-highlight-header"></div>
3   
4   <div class="caption">
5     <h2>...</h2>
6     <div class="score-highlight-body">
7       ...
8     </div>
9     <div class="score-call-to-action">
10    ...
11   </div>
12 </div>
13 </div>
```

On the right is `_site-highlights.scss`, which contains the following SCSS:

```
1 .score-highlight {
2   @extend .thumbnail; //inherit bootstrap
3   @include vertical-space($vertical-spacing);
4   .score-highlight-header {
5   }
6   img {
7   }
8   }
9   .caption {
10  h2 {
11  }
12  }
13  }
14  .score-highlight-body {
15  }
16  }
17  .score-call-to-action {
18  }
19  }
20 }
21 &.all-new-class {
22   //Class selection styles here
23 }
24 }
25 }
```

The class `all-new-class` is highlighted in both the HTML and the SCSS file, demonstrating how a class defined in the HTML can be targeted in the CSS.

# SCORE Content

Highlight with Class Selection Example:



### Rendering Transformations

Extend Sitecore's rendering compatibility to easily transform renderings.

[LEARN MORE](#)



### Rendering Transformations

Extend Sitecore's rendering compatibility to easily transform renderings.

[Learn More](#)

# LAB 5

## Exercise: Style 2 Versions of a SCORE Content Component

1. Navigate to the “\_html” folder > index.html
2. For the last two Highlight components on the page, you will create two different versions of the same component using class selections
3. Use the “icon\_grey\_rendering.png” in the “img” folder as your Highlight image



**Rendering Transformations**

Extend Sitecore's rendering compatibility to easily transform renderings.

[LEARN MORE](#)



**Rendering Transformations**

Extend Sitecore's rendering compatibility to easily transform renderings.

[Learn More](#)



8.3

# SCORE Navigation

Understanding SCORE's navigation components

# SCORE Navigation

## Navigation Components

- Navigation components create various types of UX navigation
- Navigation renderings:
  - Breadcrumb
  - Main Menu
  - Mega Menu
  - Menu List
  - My Siblings Menu
  - Section Menu Spider
  - Sitemap Spider
  - This Section Menu
- Reference: [Navigation Documentation](#)

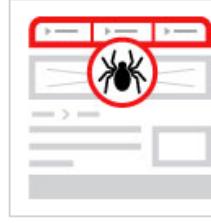
# SCORE Navigation

## Navigation Renderings for Header

Select a Rendering  
Select the rendering that you want to use. Click Select to continue.

CONTENT     NAVIGATION     PAGE LAYOUT

  
Mega Menu

  
Main Menu Spider

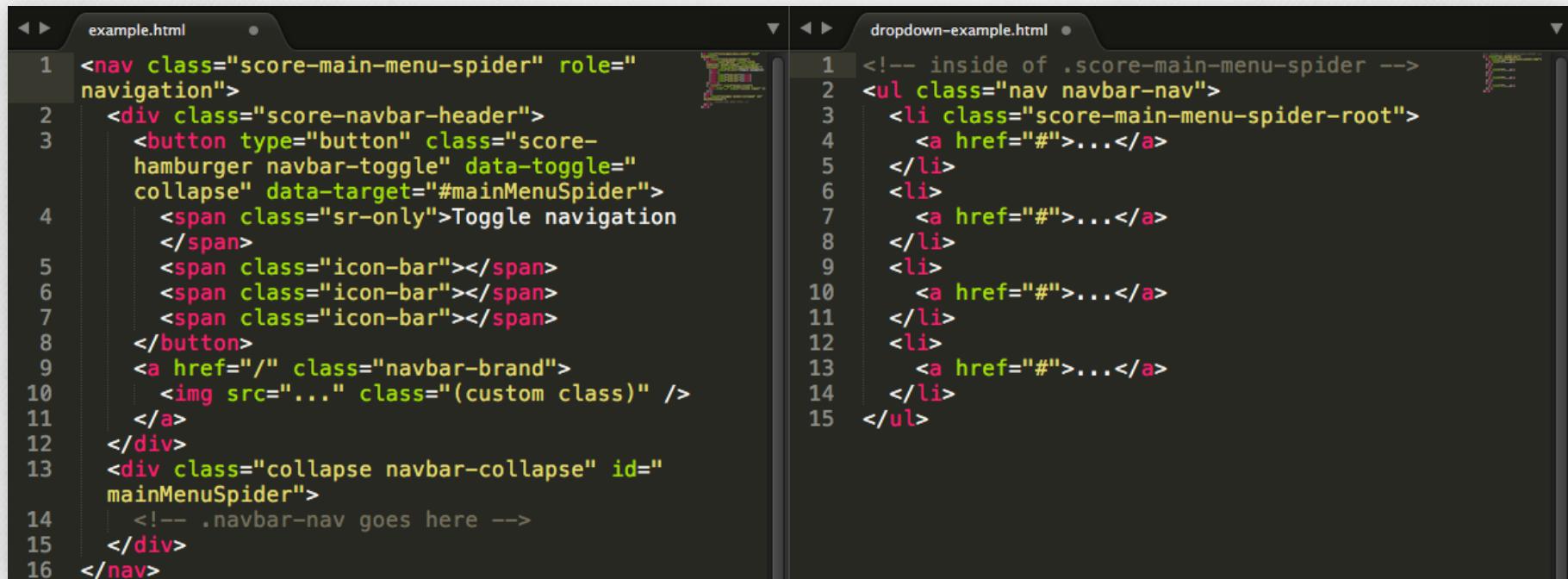
Open the Properties dialog box immediately.

Select     Cancel

# SCORE Navigation

## Main Menu Spider Example:

- For Navigation with basic links and/or basic dropdowns



The image shows a code editor with two tabs open: "example.html" and "dropdown-example.html".

**example.html:**

```
1 <nav class="score-main-menu-spider" role="navigation">
2   <div class="score-navbar-header">
3     <button type="button" class="score-hamburger navbar-toggle" data-toggle="collapse" data-target="#mainMenuSpider">
4       <span class="sr-only">Toggle navigation</span>
5       <span class="icon-bar"></span>
6       <span class="icon-bar"></span>
7       <span class="icon-bar"></span>
8     </button>
9     <a href="/" class="navbar-brand">
10       
11     </a>
12   </div>
13   <div class="collapse navbar-collapse" id="mainMenuSpider">
14     <!-- .navbar-nav goes here -->
15   </div>
16 </nav>
```

**dropdown-example.html:**

```
1 <!-- inside of .score-main-menu-spider -->
2 <ul class="nav navbar-nav">
3   <li class="score-main-menu-spider-root">
4     <a href="#">...</a>
5   </li>
6   <li>
7     <a href="#">...</a>
8   </li>
9   <li>
10    <a href="#">...</a>
11  </li>
12  <li>
13    <a href="#">...</a>
14  </li>
15 </ul>
```

# SCORE Navigation

## Mega Menu Example:

- For Navigation with basic links and/or mega menu dropdowns

```
example.html
1 <nav class="score-megamenu" role="navigation">
2   <div class="score-navbar-header">
3     <button type="button" data-toggle="collapse"
4           data-target="#menu" class="navbar-toggle">
5       <span class="icon-bar"></span>
6       <span class="icon-bar"></span>
7       <span class="icon-bar"></span>
8     </button>
9     <a href="#" class="navbar-brand">
10      
11    </a>
12  </div>
13  <div class="navbar-collapse collapse" id="menu">
14    |   <!-- navbar row component goes here -->
15  </div>
</nav>
```

```
dropdown-example.html
1 <!-- navbar row component example -->
2 <div class="score-column1">
3   <div class="score-center">
4     ...
5   </div>
6 </div>
7
8 <!-- inside of navbar row component -->
9 <ul class="nav navbar-nav score-nav">
10 ...
11 </ul>
```

# SCORE Navigation

## Navigation Renderings

Select a Rendering  
Select the rendering that you want to use. Click Select to continue.

□ X

BOOTSTRAPUI TEST    BUTTONS    COLLECTIONS    CONTENT    DOCUMENT HEADERS    FEATURES

INNER STRUCTURE    NAVIGATION    PANELS    SECTION HEADERS    SOCIAL

Menu List    Section Menu Spider    Breadcrumb    Previous Next Siblings Menu    My Siblings Menu

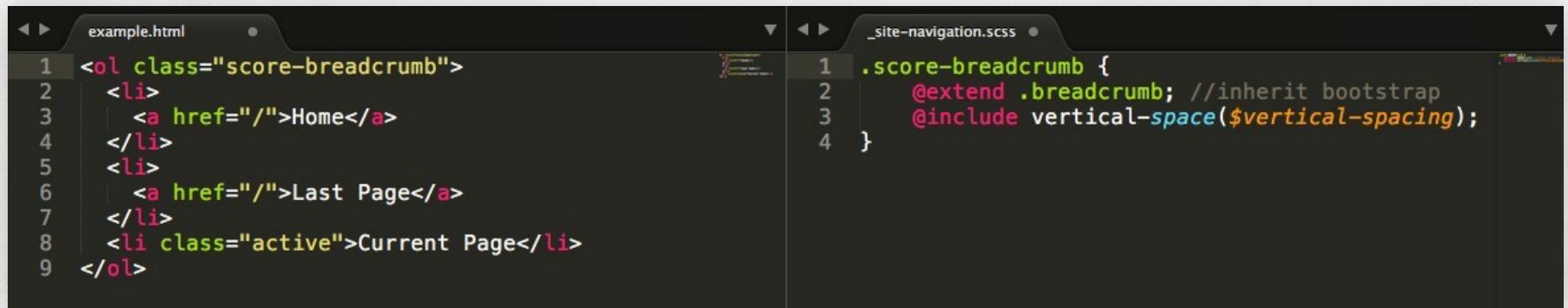
This Section Menu    Siteman Spider

Open the Properties dialog box immediately.

Select    Cancel

# SCORE Navigation

Breadcrumb Example:



The screenshot shows a code editor with two files side-by-side. On the left is `example.html`, which contains the following HTML code:

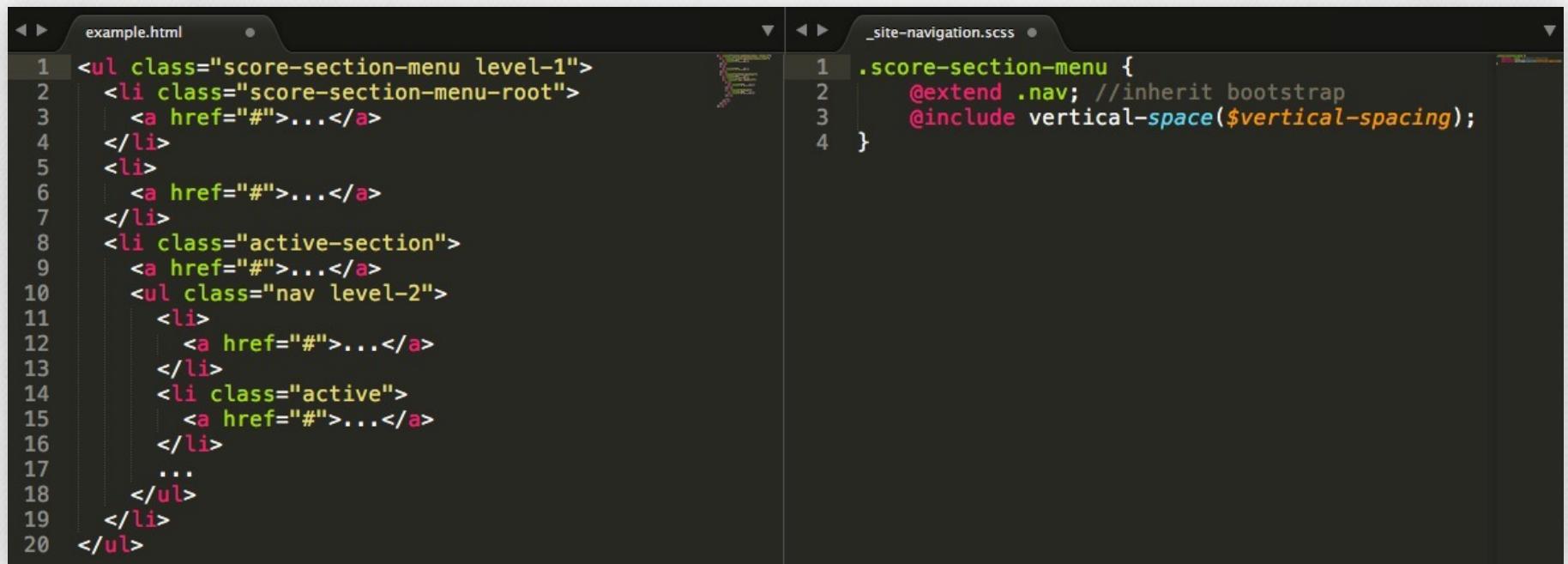
```
1 <ol class="score-breadcrumb">
2   <li>
3     <a href="/">Home</a>
4   </li>
5   <li>
6     <a href="/">Last Page</a>
7   </li>
8   <li class="active">Current Page</li>
9 </ol>
```

On the right is `_site-navigation.scss`, which contains the following SCSS code:

```
1 .score-breadcrumb {
2   @extend .breadcrumb; //inherit bootstrap
3   @include vertical-space($vertical-spacing);
4 }
```

# SCORE Navigation

## Section Menu Example:



The image shows a code editor interface with two files open: `example.html` and `_site-navigation.scss`.

`example.html` contains the following HTML code:

```
1 <ul class="score-section-menu level-1">
2   <li class="score-section-menu-root">
3     <a href="#">...</a>
4   </li>
5   <li>
6     <a href="#">...</a>
7   </li>
8   <li class="active-section">
9     <a href="#">...</a>
10    <ul class="nav level-2">
11      <li>
12        <a href="#">...</a>
13      </li>
14      <li class="active">
15        <a href="#">...</a>
16      </li>
17      ...
18    </ul>
19  </li>
20 </ul>
```

`_site-navigation.scss` contains the following SCSS code:

```
1 .score-section-menu {
2   @extend .nav; //inherit bootstrap
3   @include vertical-space($vertical-spacing);
4 }
```

# Exercise: Build and Style Main Navigation

LAB  
6

1. Navigate to <http://www.sugcon.com>
2. Identify the SCORE navigation components used in the main navigation
3. Build navigation in Sitecore (this part will be a demo with your assistance!)
4. Style the navigation using Sass (this part will be a demo with your assistance!)



8.3

# SCORE Social

SCORE's social sharing component

# SCORE Social



- [Add This Plugin](#)



Facebook Like Button



Facebook Send Button



Twitter Tweet Button



Twitter Follow Button



Google+ Badge



LinkedIn Button



Reddit Button



Pinterest Pin-It Button

09

## SCORE 2.0 Migration

Migration of existing sites or  
starting fresh with 2.0 UI

# SCORE 2.0 Migration



## 3 Scenarios:

- New Site(s)
- Support Existing Site w/ LESS
- Support Existing Site w/ Plain CSS
- Reference: [Migration Documentation](#)

# Thank you

Ben Cato

[ben@brainjocks.com](mailto:ben@brainjocks.com)

Emily Lord

[emily@brainjocks.com](mailto:emily@brainjocks.com)

