BRAINJOCKS SCORE™

SCORE 2.0 UI

Ben Cato & Emily Lord

BRAINJOCKS SCORE™ 2.0

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

01

The Value of SCORE

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

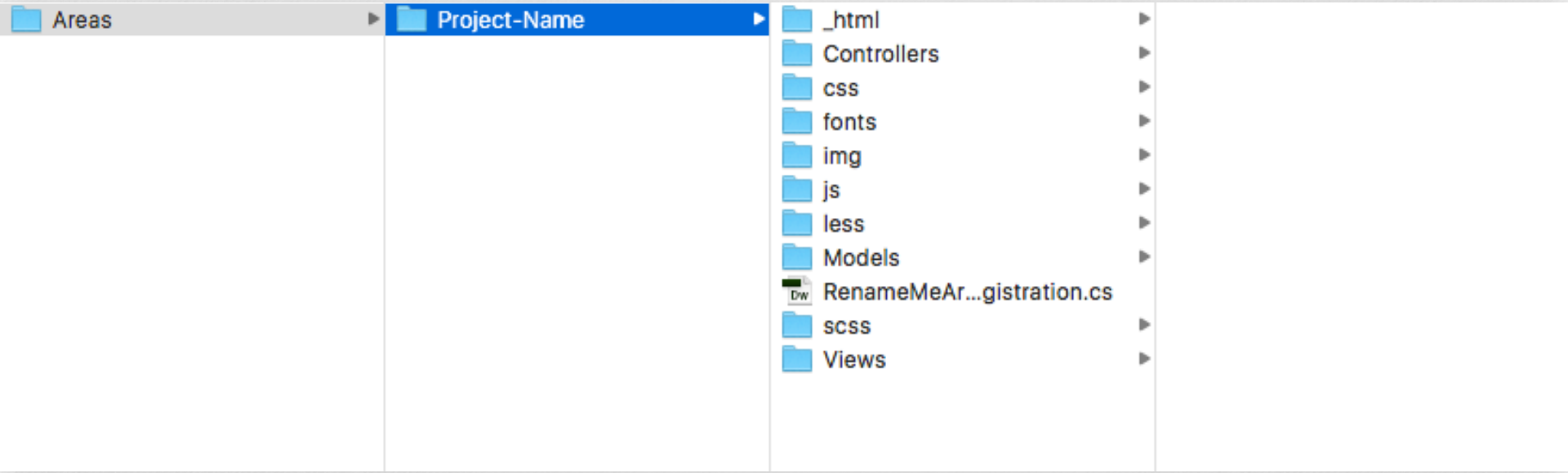
"Areas" Scaffolding

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

File Structure

"RenameMeArea" Folder

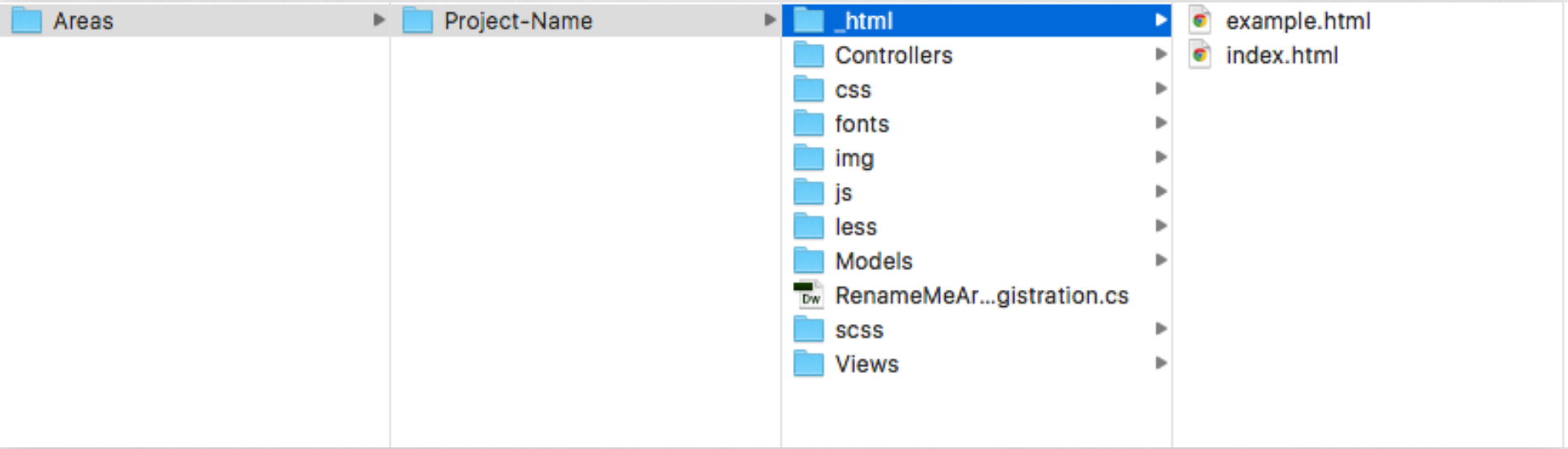
- Renamed to a specific project/site name



File Structure

HTML “Sandbox” folder

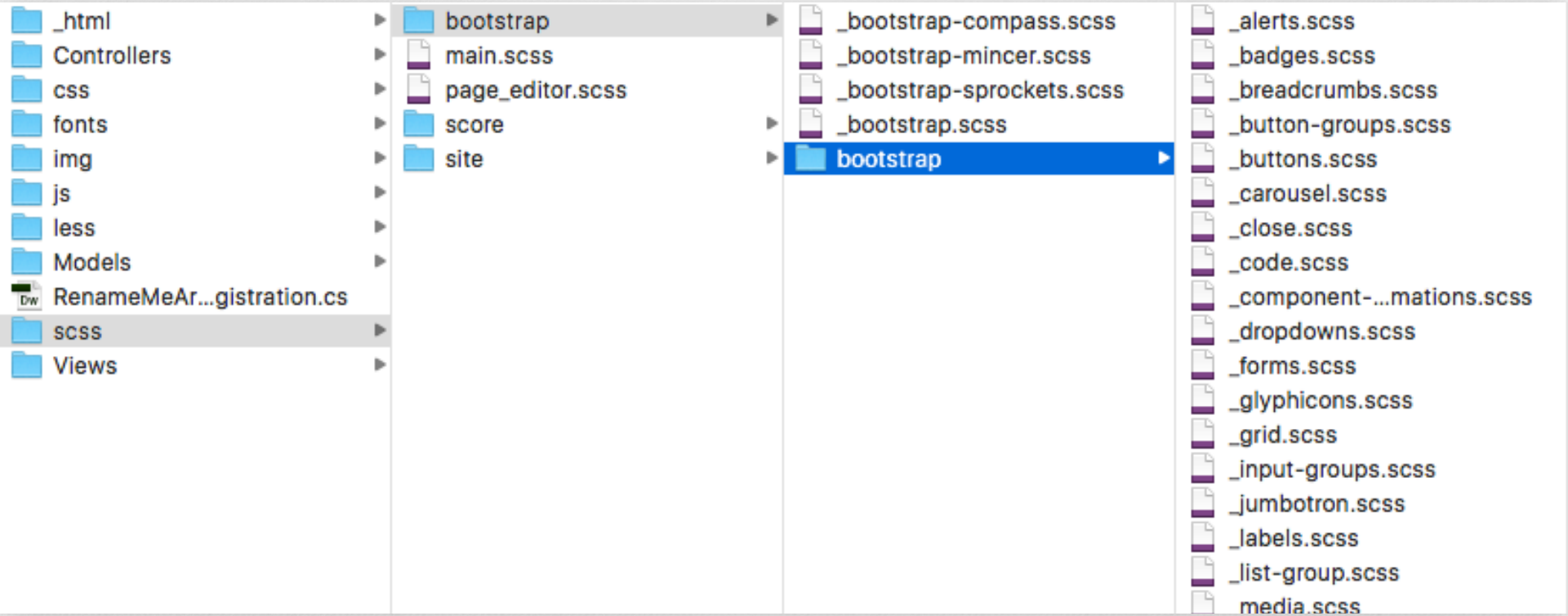
- Local HTML that is for your use - ***Does not get deployed***
- example.html provided for “kitchen sink” examples
- index.html “boilerplate” provided for start-up



File Structure

Bootstrap Folder

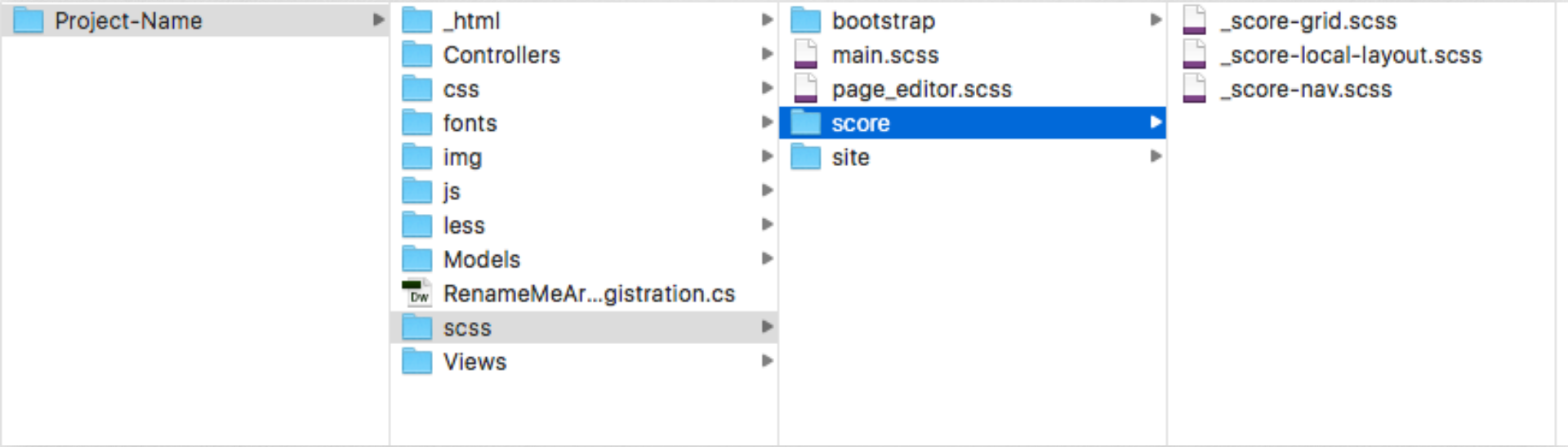
- Contains all of Bootstrap's default Sass files
- Location to access Bootstrap's variables and mixins (if needed)



File Structure

SCORE Folder

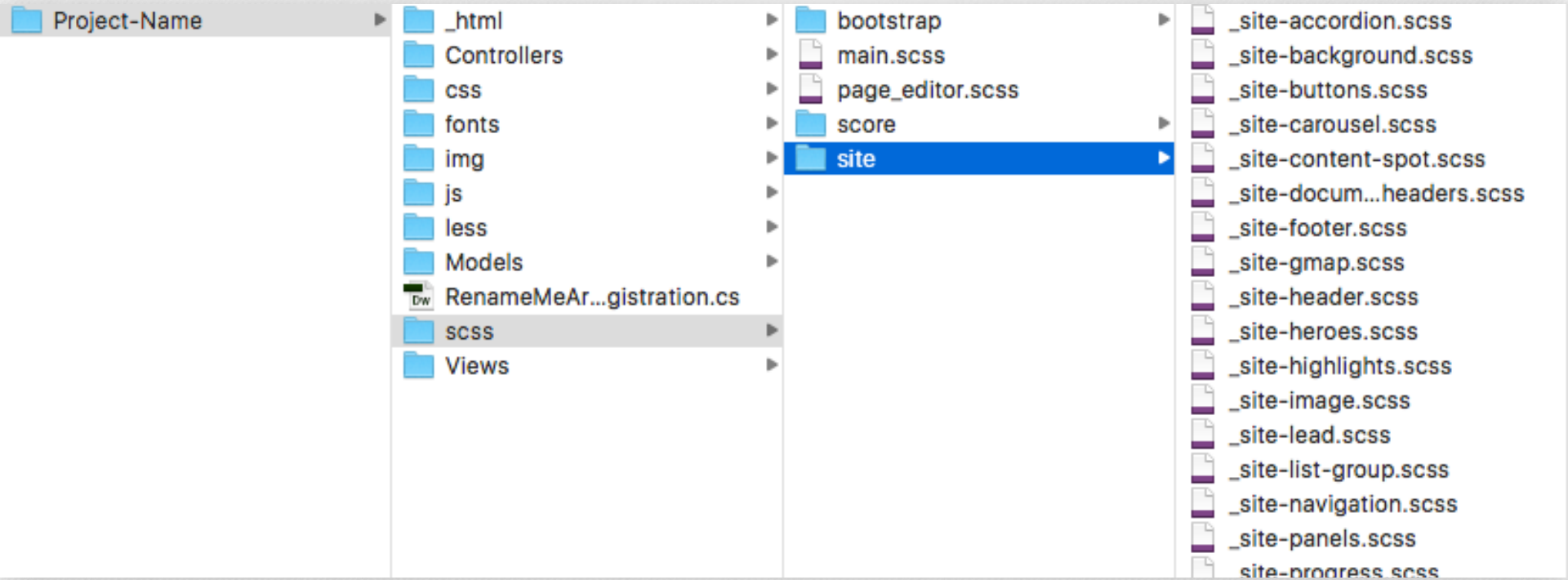
- Contains all of SCORE's default Sass files
- Location to modify SCORE-specific styles (if needed)



File Structure

Site Folder

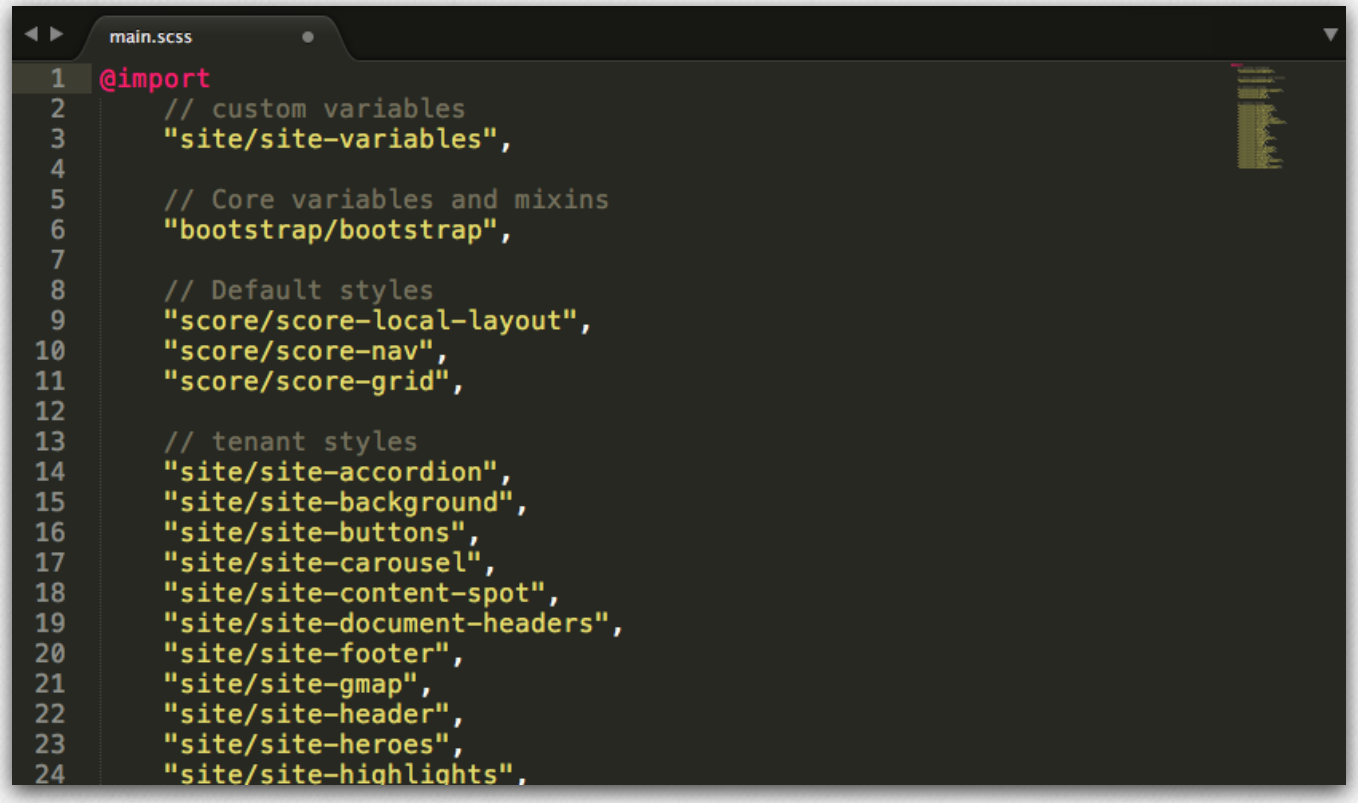
- Location to modify/include all site-specific styles
- Create site-specific variables and mixins



File Structure

main.scss

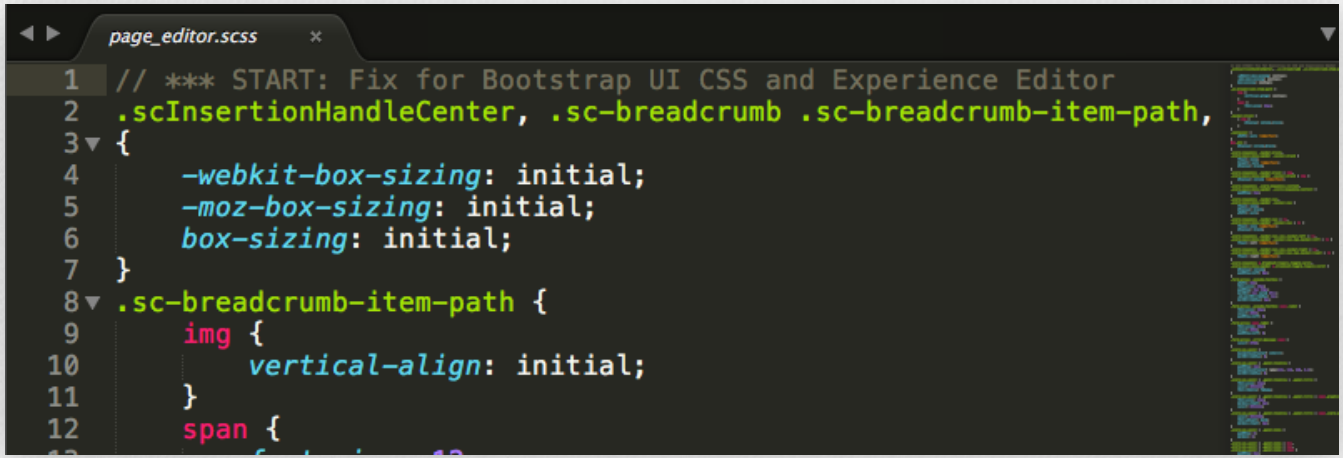
- main.scss imports all Bootstrap, SCORE, and site Sass files
- Compiles to main.css



File Structure

page_editor.scss

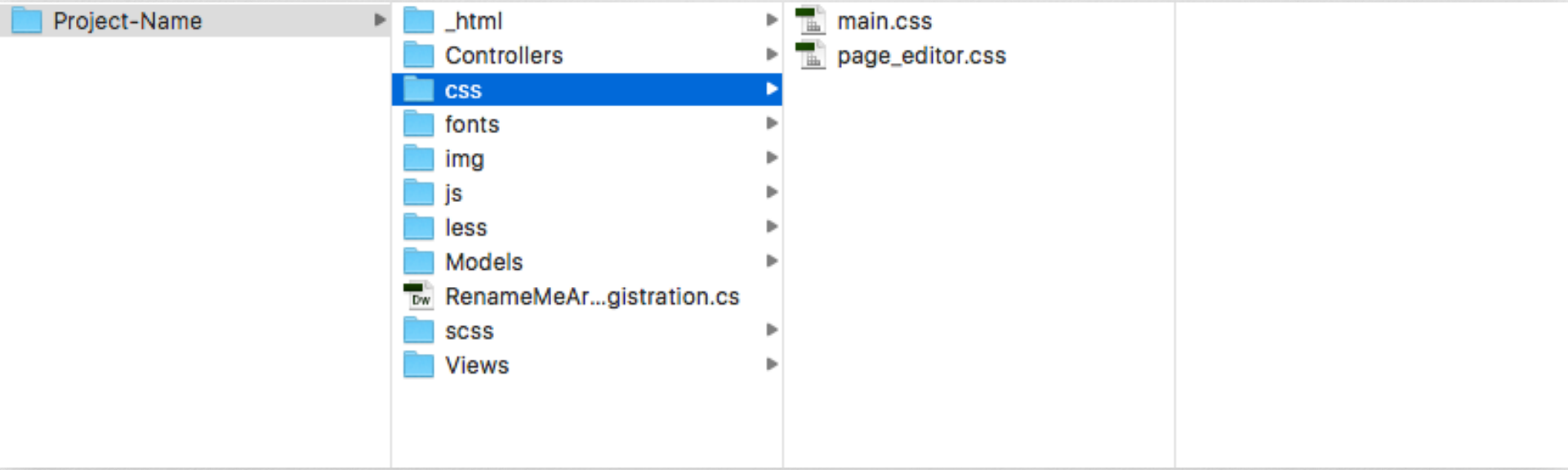
- Styles are specific to the SCORE experience editor
- Components may need to render differently in experience editor via CSS
- Components initially inherit main.css
- page_editor.css ***overrides*** main.css and can include additional styles if needed



File Structure

Project CSS files

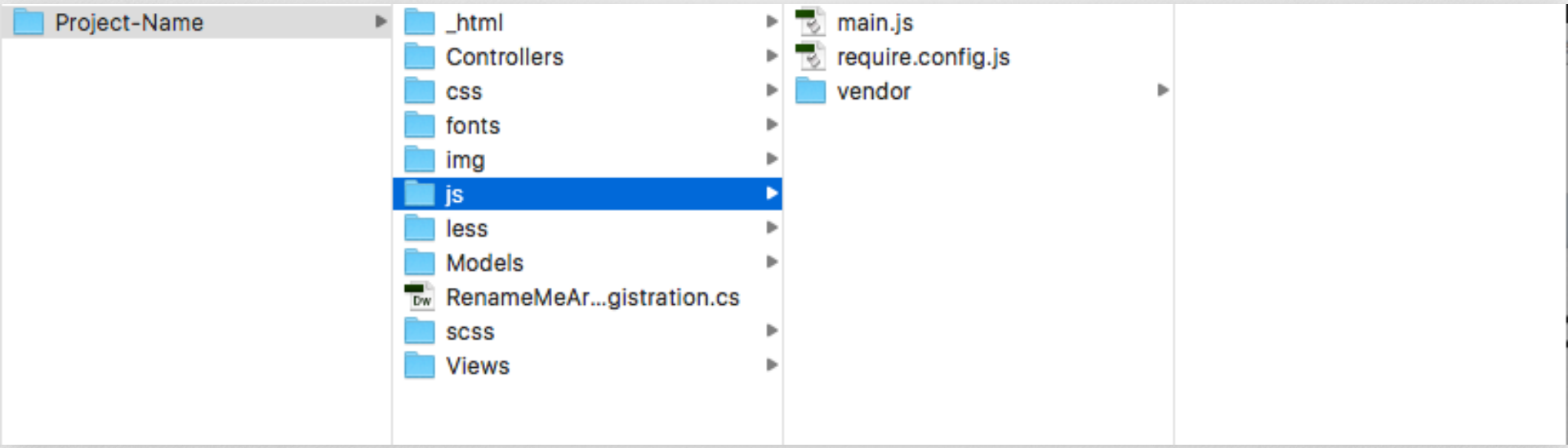
- ***NEVER*** edit CSS styles directly
- CSS files are ***ONLY*** to be compiled via Sass



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

index.html

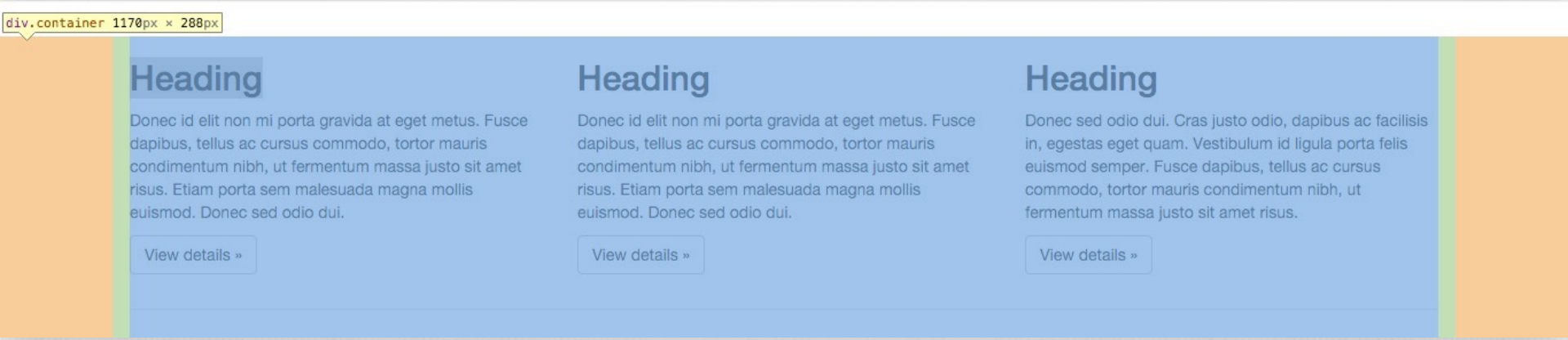
```
<div class="container">
  <div class="row">
    <div class="col-md-4">4 column</div>
    <div class="col-md-4">4 column</div>
    <div class="col-md-4">4 column</div>
  </div>
  <div class="row">
    <div class="col-md-6">6 column</div>
    <div class="col-md-6">6 column</div>
  </div>
</div>
```

- 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)



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

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

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**

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

Mobile Breakpoints

Bootstrap

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 »

Desktop

Mobile

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

index.html

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

Bootstrap

Column Operations

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

index.html

```
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>
```

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 (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
.visible-xs-*	Visible	Hidden	Hidden	Hidden
.visible-sm-*	Hidden	Visible	Hidden	Hidden
.visible-md-*	Hidden	Hidden	Visible	Hidden
.visible-lg-*	Hidden	Hidden	Hidden	Visible
.hidden-xs	Hidden	Visible	Visible	Visible
.hidden-sm	Visible	Hidden	Visible	Visible
.hidden-md	Visible	Visible	Hidden	Visible
.hidden-lg	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:
    -
    -
    -

- 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

07

Styling & Inheritance with Sass

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

Compiler

_sass.scss

```scss
//Sass
ul {
    list-style: none;
    li {
        float: left;
        a {
            color: #ff0000;
        }
    }
}
```

Compiler

_sass.css

```css
ul {
    list-style: none;
}
ul li {
    float: left;
}
ul li a {
    color: #ff0000;
}
```

Sass

Command Line

Mac

Windows

- Reference: Installing a compiler

Exercise: Installing a Compiler

LAB 1

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

main.scss _example.scss

```scss
//Sass
ul {
    list-style: none;
    li {
        float: left;
        margin-right: 10px;
        a {
            color: #ff0000;
        }
    }
}
```

main.css

```css
ul {
  list-style: none;
}
ul li {
  float: left;
  margin-right: 10px;
}
ul li a {
  color: #ff0000;
}

```

Sass

Bad Nesting Example

main.scss _example.scss

```scss
.container {
    .row-class {
        .column-class {
            ul {
                list-style: none;
                li {
                    float: left;
                    margin-right: 10px;
                    a {
                        color: #ff0000;
                    }
                }
            }
        }
    }
}
```

main.css

```css
.container .row-class .column-class ul {
    list-style: none;
}
.container .row-class .column-class ul li {
    float: left;
    margin-right: 10px;
}
.container .row-class .column-class ul li a {
    color: #ff0000;
}
```

Sass

Nesting Media Queries

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

main.scss / _example.scss

```scss
//Sass
.class-1 {
    @media only screen and (min-width: 768px) {
        color: #ff0000;
    }
    @media only screen and (min-width: 992px) {
        color: #2b2b2b;
    }
}
```

main.css

```css
@media only screen and (min-width: 768px) {
  .class-1 {
    color: #ff0000;
  }
}
@media only screen and (min-width: 992px) {
  .class-1 {
    color: #2b2b2b;
  }
}

```

Sass

Tying Up The Markup & Styling

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

index.html

```
<div class="score-highlight">
  <div class="score-highlight-header">
    ...
  </div>
  <img src="..." alt="#">
  <div class="caption">
    <h2>...</h2>
    <div class="score-highlight-body">
      ...
    </div>
    <div class="score-call-to-action">
      ...
    </div>
  </div>
</div>
```

_site-highlights.scss

```
.score-highlight {
    @extend .thumbnail; //inherit bootstrap
    @include vertical-space($vertical-spacing);
    .score-highlight-header {

    }
    img {

    }
    .caption {
        h2 {

        }
        .score-highlight-body {

        }
        .score-call-to-action {

        }
    }
}

```

Sass

Sass Operators

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

_sass.scss

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

css.css

```
.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

_sass.scss

```scss
h2 {
    font-size: 20px;
    font-weight: bold;
    color: #ff0000;
    .parent-class1 & {
        font-size: 24px;
        font-weight: normal;
    }
}
```

_site-navigation.scss | css.css

```css
h2 {
    font-size: 20px;
    font-weight: bold;
    color: #ff0000;
}
.parent-class1 h2 {
    font-size: 24px;
    font-weight: normal;
}
```

Sass

Sass Operators

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

_sass.scss

```
1  //Sass
2  .class1 {
3      & > .class2 {
4          color: #ff0000;
5      }
6  }
```

css.css

```
1  .class1 > .class2 {
2      color: #ff0000;
3  }
```

Sass

Extending Classes

- @extend adds the current selector to the selector that is being extended

- Reference: [The extend concept]

_sass.scss

```
1  //Sass
2  .blue {
3      color: blue;
4  }
5  nav ul {
6      @extend .blue; //inherit .blue class
7      padding: 0;
8  }
```

css.css

```
1  .blue, nav ul {
2      color: blue;
3  }
4  nav ul {
5      padding: 0;
6  }
```

7.2

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

```
1  .score-highlight {
2      @extend .thumbnail; //inherit bootstrap
3  }
4
```

_main.css

```
5910
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;
```

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.)

index.html

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

site-highlight.less

```
.score-highlight {
    //overriding Bootstrap styles went here
    ...
    ...
    ...
    ...
}
```

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

index.html

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

_site-highlights.scss

```
1 .score-highlight {
2     @extend .thumbnail; //inherit bootstrap
3 }
4
```

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

main.scss

```scss
//Sass variables
$brand-color: #ff0000;

//Sass styles
header, footer {
    background-color: $brand-color;
}
h1 {
    color: $brand-color;
}
```

_example.scss

main.css

```css
header, footer {
  background-color: red;
}

h1 {
  color: red;
}

```

Exercise: Using a Variable

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

```scss
//Global custom variables

//default margin on content components
$vertical-spacing: 30px;

//Global custom mixins
@mixin vertical-space($margin) {
    margin-bottom: $margin;
}
@mixin rounded-corners($radius) {
    -webkit-border-radius: $radius;
    -moz-border-radius: $radius;
    -0-border-radius: $radius;
    border-radius: $radius;
}
```

_site-highlights.scss

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

```

Sass

Sass Mixins

- Example of pixel to rem converter mixin

_sass.scss

```scss
@mixin pxConvert($px) {
    font-size: ($px / 16) + rem;
}

h2 {
    @include pxConvert(12);
    font-weight: bold;
    color: #ff0000;
}
```

css.css

```css
h2 {
  font-size: 0.75rem;
  font-weight: bold;
  color: #ff0000;
}
```

Sass

Vertical Spacing

- Set to 30px for vertical space by default

_site-variables.scss

```
//Global custom variables

//default margin on content components
$vertical-spacing: 30px;

//Global custom mixins
@mixin vertical-space($margin) {
    margin-bottom: $margin;
}
```

_site-highlights.scss

```
.score-highlight {
    @extend .thumbnail; //inherit bootstrap
    @include vertical-space($vertical-spacing);
    .score-highlight-header {

    }
    img {

    }
    .caption {
        h2 {

        }
        .score-highlight-body {

        }
        .score-call-to-action {

        }
    }
}

```

Exercise: Using a Mixin

LAB 3

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

08

SCORE Components

8.1

SCORE Containers

Understanding SCORE's grid system

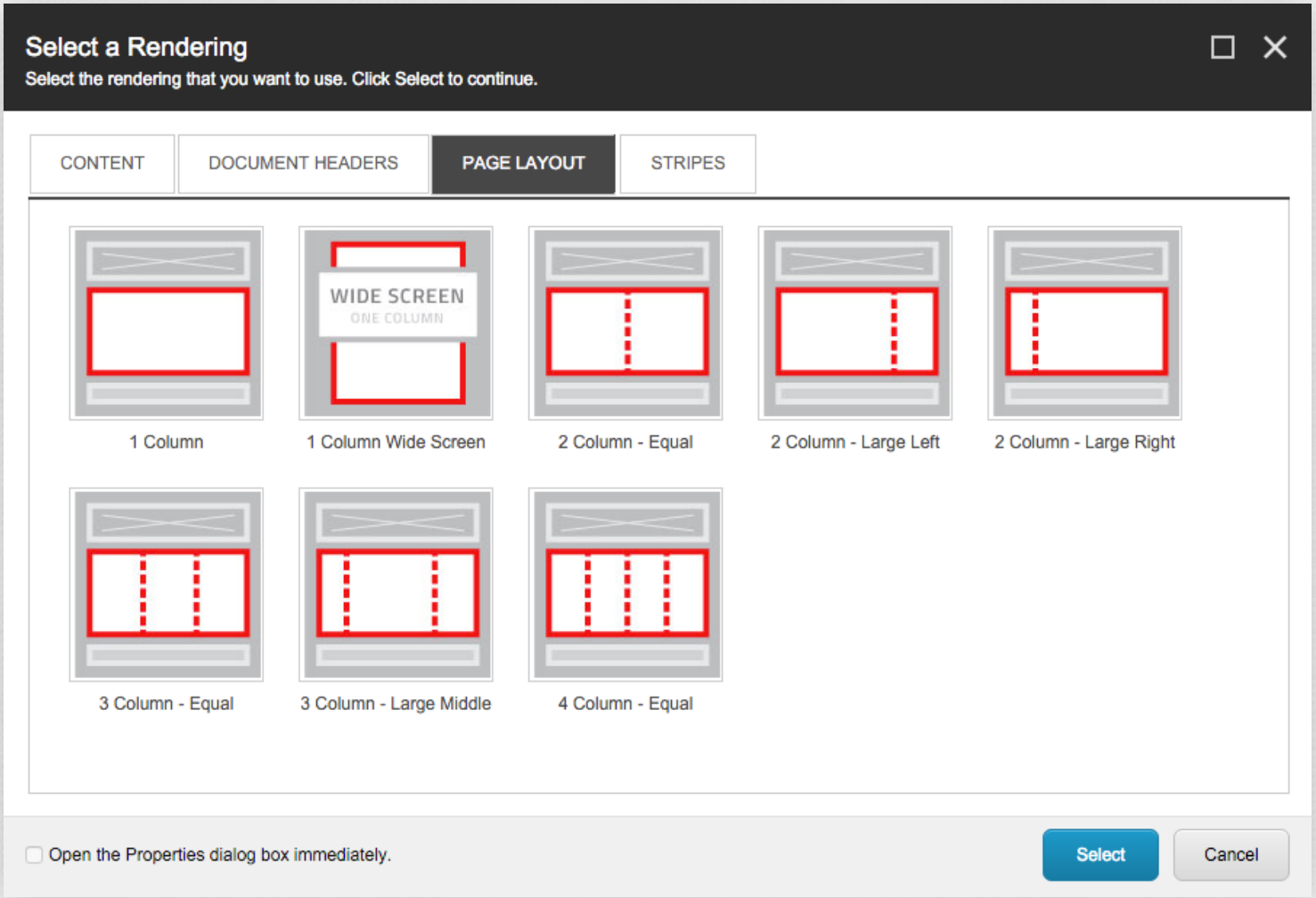
SCORE Containers

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



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

bootstrap.html

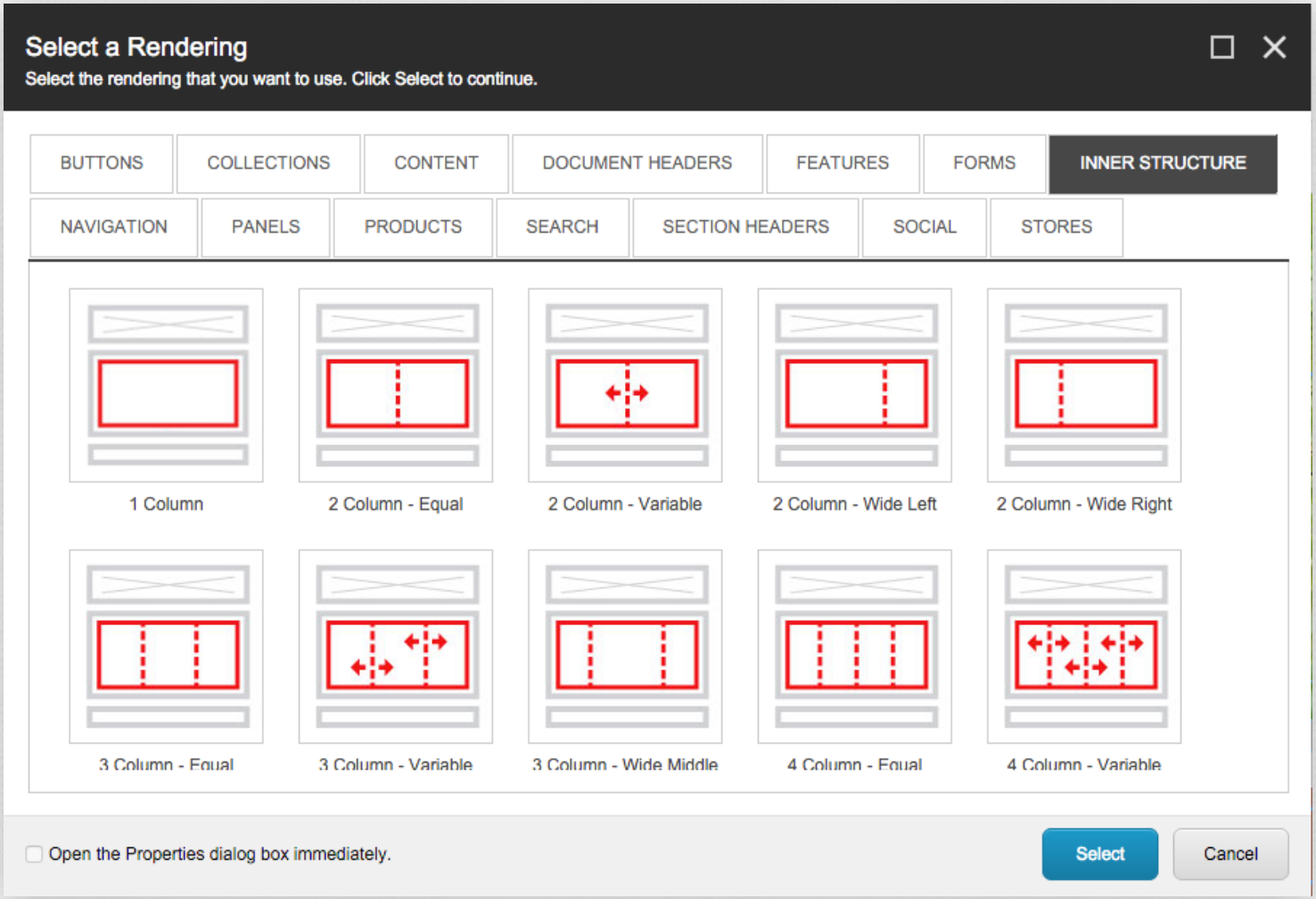
```
<!-- Bootstrap row with columns -->
<div class="container">
  <div class="row">
    <div class="col-md-12">
      ...
    </div>
  </div>
</div>
```

score.html

```
<!-- SCORE one column -->
<div class="container">
  <div class="score-column1">
    <div class="score-center">
      ...
    </div>
  </div>
</div>
```

SCORE Containers

Inner Structure



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

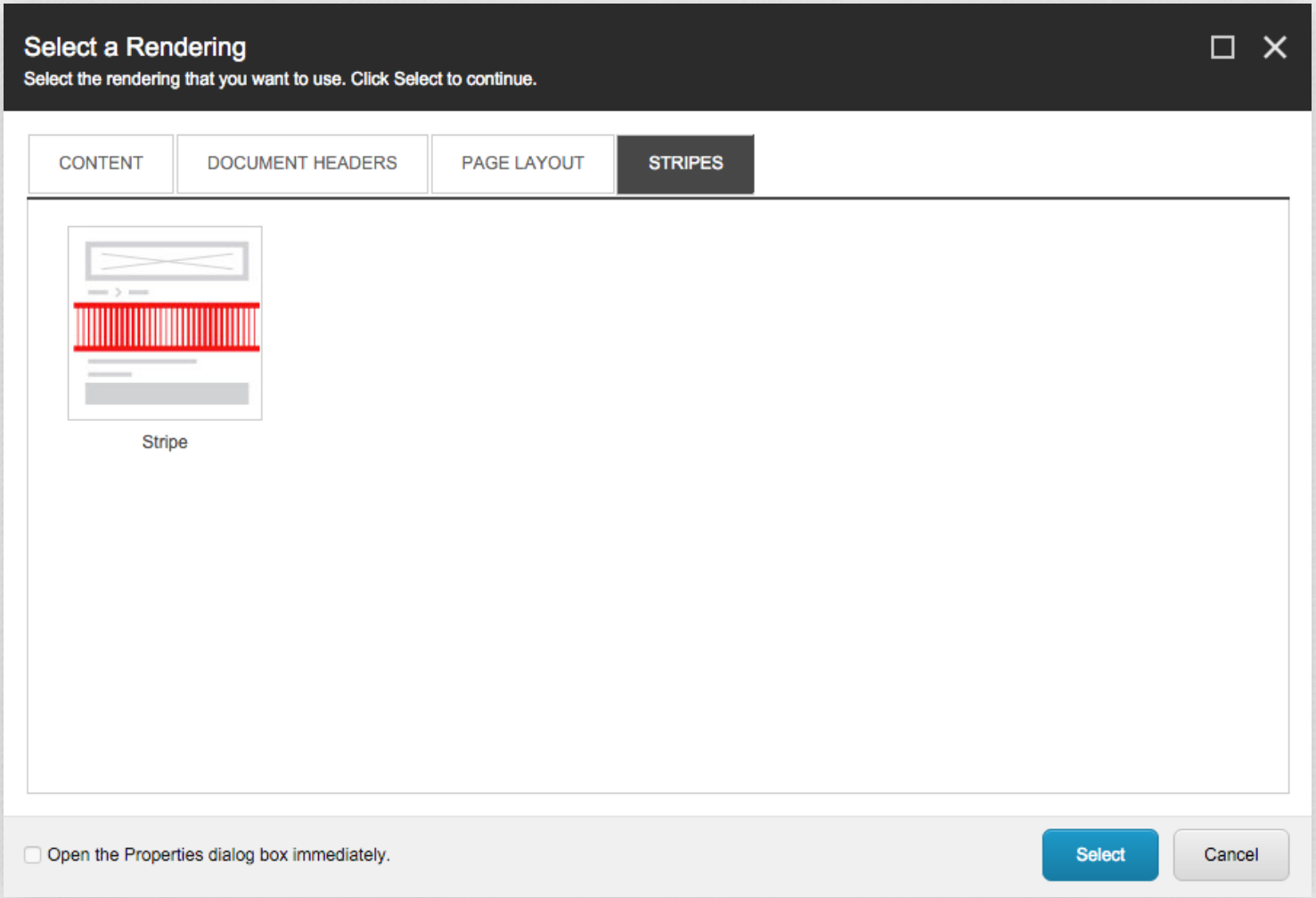
```
<!-- Bootstrap row with nested columns -->
<div class="container">
  <div class="row">
    <div class="col-md-12">
      <div class="row">
        <div class="col-md-6">
          ...
        </div>
        <div class="col-md-6">
          ...
        </div>
      </div>
    </div>
  </div>
</div>
```

score.html

```
<!-- SCORE row with nested columns -->
<div class="container">
  <div class="score-column1">
    <div class="score-center">
      <div class="score-column2 equal">
        <div class="score-left">
          ...
        </div>
        <div class="score-right">
          ...
        </div>
      </div>
    </div>
  </div>
</div>
```

SCORE Containers

Stripe



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

```
<!-- Bootstrap full width wrapper -->
<div class="full-width-wrapper">
  <div class="container">
    <div class="row">
      <div class="col-md-12">
        ...
      </div>
    </div>
  </div>
</div>
```

score.html

```
<!-- SCORE stripe -->
<div class="score-stripe">
  <div class="container">
    <div class="score-column1">
      <div class="score-center">
        ...
      </div>
    </div>
  </div>
</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

```scss
//set up grid size
$grid-columns: 12 !default;

//turn any class with .score-column into a row
div[class^="score-column"], div[class*=" score-column"] {
    @include make-row();
    &.score-column1 {
        & > .score-center {
            @include make-md-column($grid-columns);
            @media print {
                @include make-xs-column($grid-columns);
            }
        }
    }
    &.score-column2 {
        &.equal {
            & > .score-left {
                @include make-md-column($grid-columns/2);
                @media print {
                    @include make-xs-column($grid-columns/2);
                }
            }
            & > .score-right {
                @include make-md-column($grid-columns/2);
                @media print {
                    @include make-xs-column($grid-columns/2);
                }
```

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!

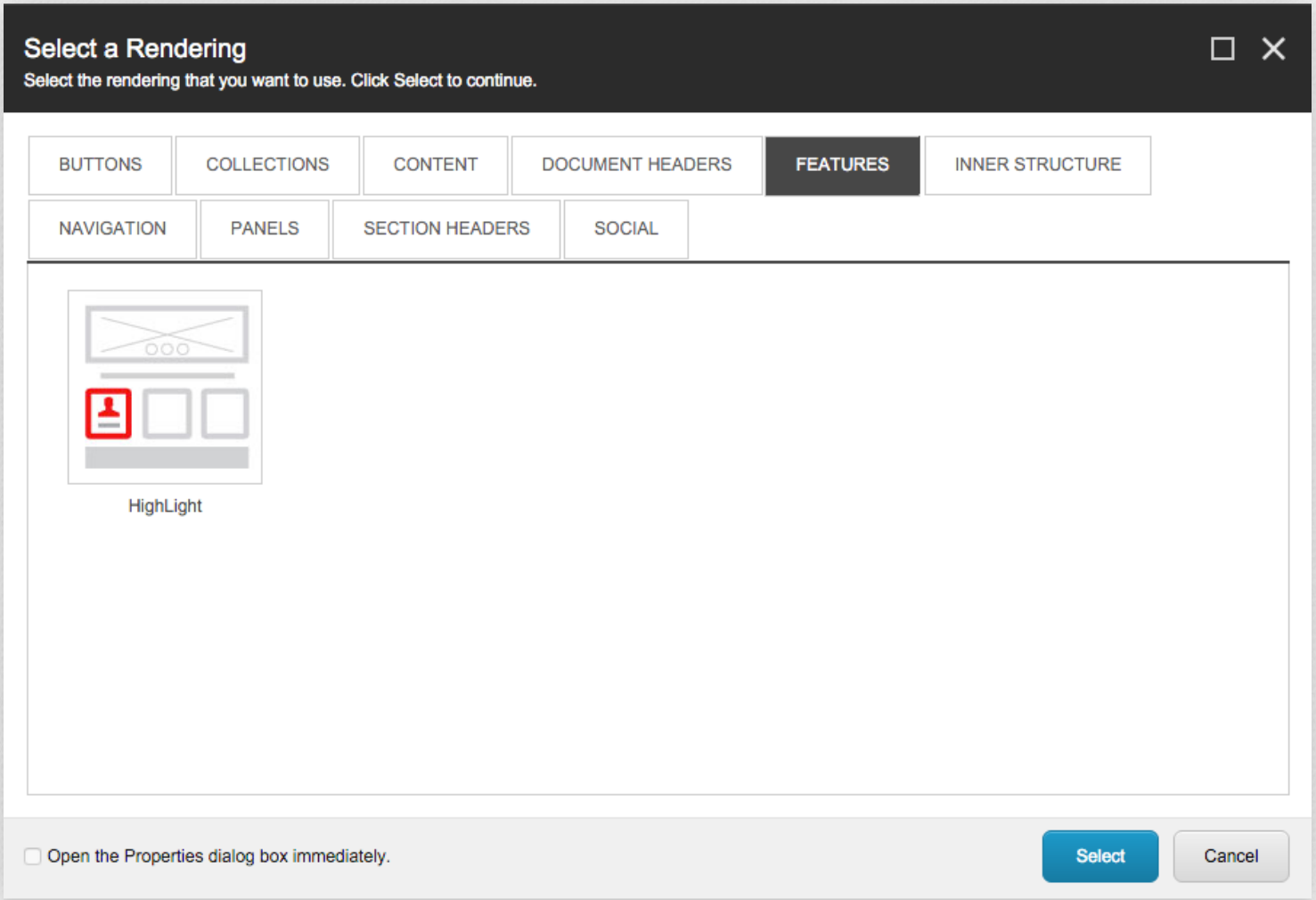
- The key to gaining full advantage of SCORE is all about ***reuse***.
  - Familiarize yourself with every SCORE component
  - Know when/how/where to use default components
  - Saves you time and money!

- 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



SCORE Content

Highlight Example:

- Generated HTML on left - Stylesheet on right

example.html

```html
<div class="score-highlight">
  <div class="score-highlight-header"></div>
  <img src="..." alt="#">
  <div class="caption">
    <h2>...</h2>
    <div class="score-highlight-body">
      ...
    </div>
    <div class="score-call-to-action">
      ...
    </div>
  </div>
</div>
```

_site-highlights.scss

```scss
.score-highlight {
    @extend .thumbnail; //inherit bootstrap
    @include vertical-space($vertical-spacing);
    .score-highlight-header {

    }
    img {

    }
    .caption {
        h2 {

        }
        .score-highlight-body {

        }
        .score-call-to-action {

        }
    }
}

```

SCORE Content

Highlight with Class Selection Example:

- Generated HTML on left - Stylesheet on right

example.html

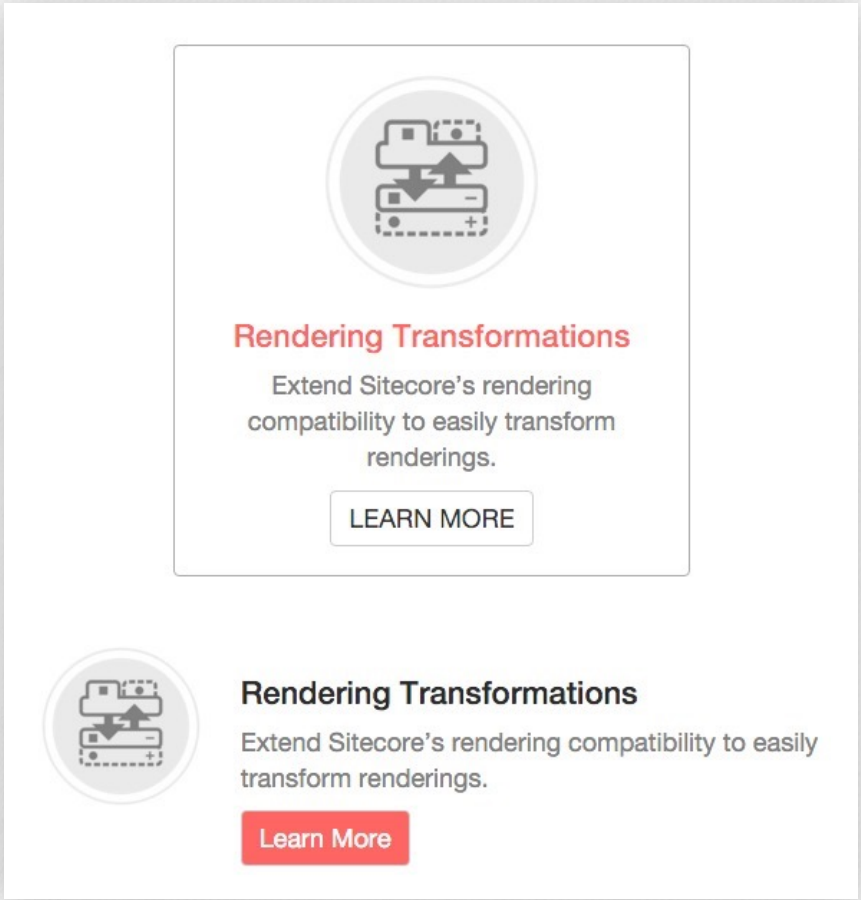
```
<div class="score-highlight all-new-class">
  <div class="score-highlight-header"></div>
  <img src="..." alt="#">
  <div class="caption">
    <h2>...</h2>
    <div class="score-highlight-body">
      ...
    </div>
    <div class="score-call-to-action">
      ...
    </div>
  </div>
</div>
```

_site-highlights.scss

```
.score-highlight {
    @extend .thumbnail; //inherit bootstrap
    @include vertical-space($vertical-spacing);
    .score-highlight-header {

    }
    img {

    }
    .caption {
        h2 {

        }
        .score-highlight-body {

        }
        .score-call-to-action {

        }
    }
    &.all-new-class {
        //Class selection styles here
    }
}

```

SCORE Content

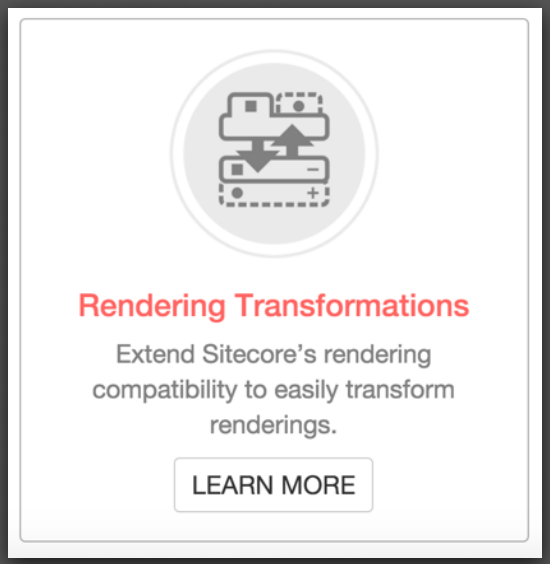
Highlight with Class Selection Example:

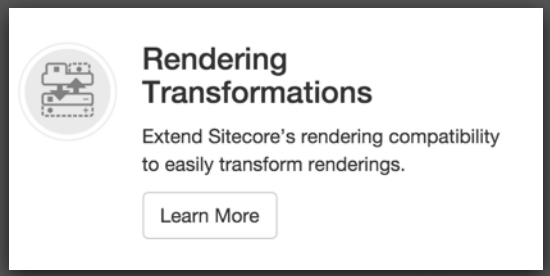


Exercise: Style 2 Versions of a SCORE Content Component

LAB 5

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





8.3

SCORE Navigation

Understanding SCORE's navigation components

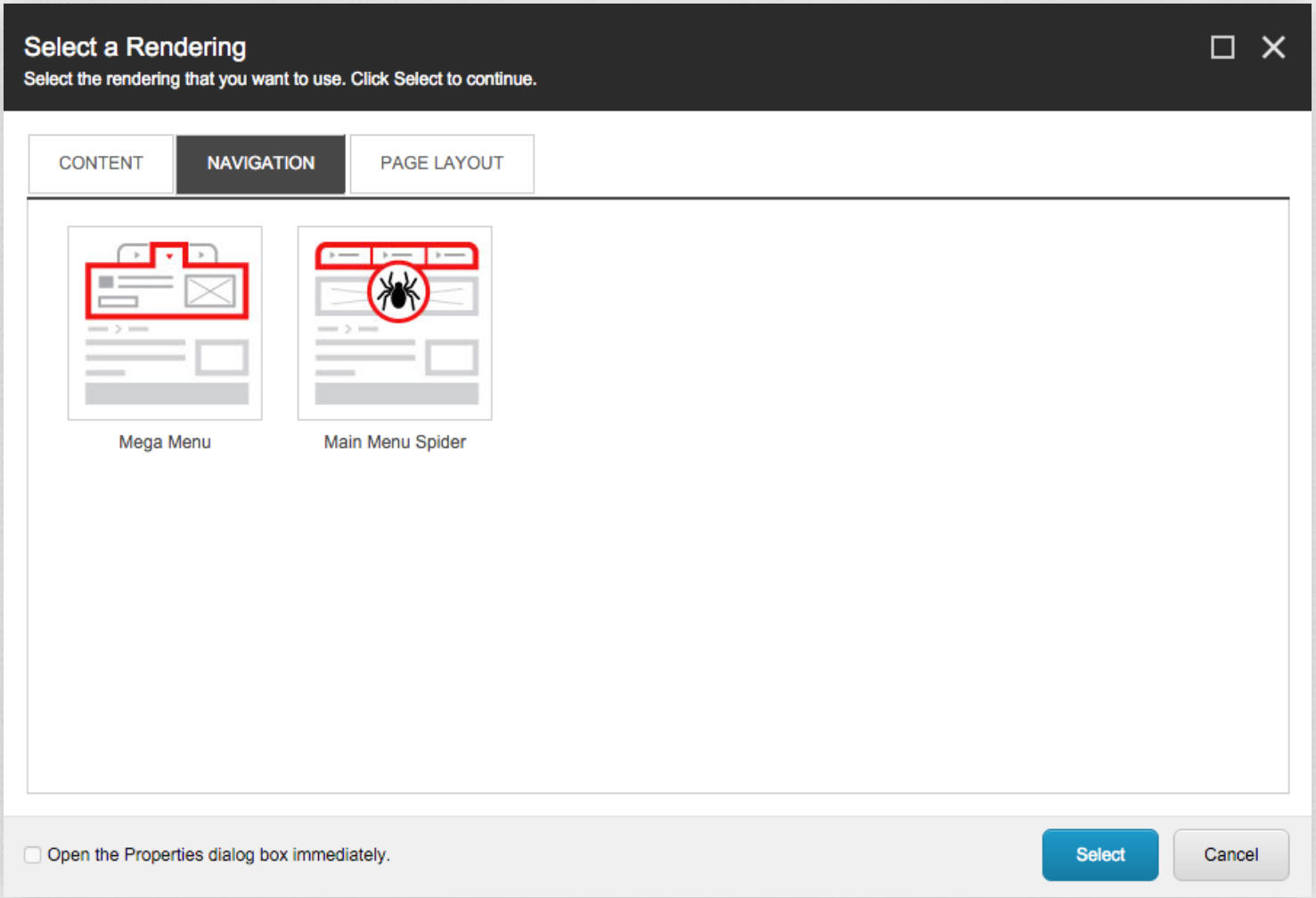
SCORE Navigation

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



SCORE Navigation

Main Menu Spider Example:

- For Navigation with basic links and/or basic dropdowns

example.html

```
<nav class="score-main-menu-spider" role="
navigation">
  <div class="score-navbar-header">
    <button type="button" class="score-
    hamburger navbar-toggle" data-toggle="
    collapse" data-target="#mainMenuSpider">
      <span class="sr-only">Toggle navigation
      </span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </button>
    <a href="/" class="navbar-brand">
      <img src="..." class="(custom class)" />
    </a>
  </div>
  <div class="collapse navbar-collapse" id="
  mainMenuSpider">
    <!-- .navbar-nav goes here -->
  </div>
</nav>
```

dropdown-example.html

```
<!-- inside of .score-main-menu-spider -->
<ul class="nav navbar-nav">
  <li class="score-main-menu-spider-root">
    <a href="#">...</a>
  </li>
  <li>
    <a href="#">...</a>
  </li>
  <li>
    <a href="#">...</a>
  </li>
  <li>
    <a href="#">...</a>
  </li>
</ul>
```

SCORE Navigation

Mega Menu Example:

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

example.html

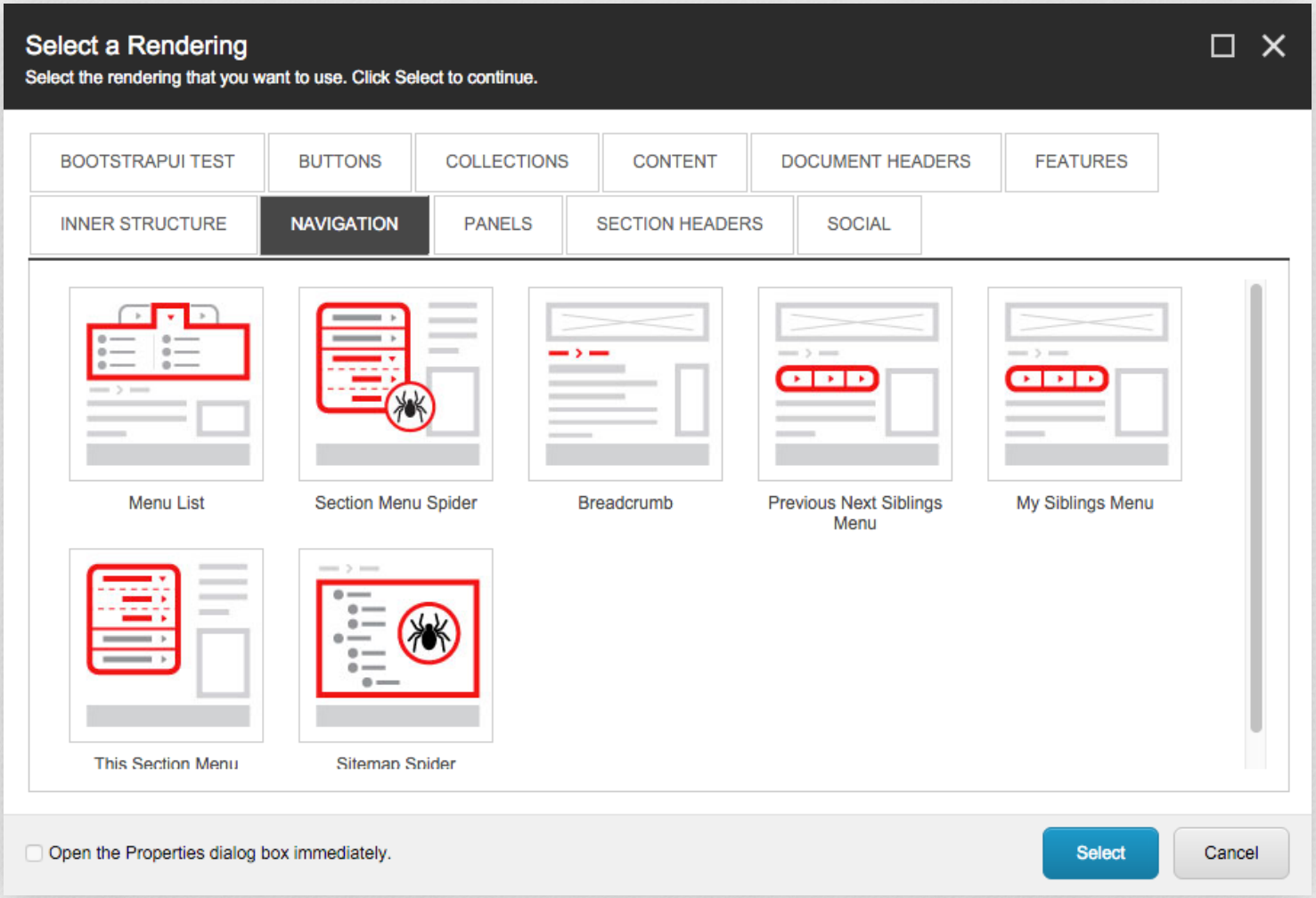
```
<nav class="score-megamenu" role="navigation">
  <div class="score-navbar-header">
    <button type="button" data-toggle="collapse"
     data-target="#menu" class="navbar-toggle">
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </button>
    <a href="#" class="navbar-brand">
      <img src="..." class="(custom class)" />
    </a>
  </div>
  <div class="navbar-collapse collapse" id="menu
  ">
    <!-- navbar row component goes here -->
  </div>
</nav>
```

dropdown-example.html

```
<!-- navbar row component example -->
<div class="score-column1">
  <div class="score-center">
    ...
  </div>
</div>


<!-- inside of navbar row component -->
<ul class="nav navbar-nav score-nav">
  ...
</ul>
```

SCORE Navigation

Navigation Renderings



SCORE Navigation

Breadcrumb Example:

example.html

```html
<ol class="score-breadcrumb">
  <li>
    <a href="/">Home</a>
  </li>
  <li>
    <a href="/">Last Page</a>
  </li>
  <li class="active">Current Page</li>
</ol>
```

_site-navigation.scss

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

SCORE Navigation

Section Menu Example:

example.html

```html
<ul class="score-section-menu level-1">
  <li class="score-section-menu-root">
    <a href="#">...</a>
  </li>
  <li>
    <a href="#">...</a>
  </li>
  <li class="active-section">
    <a href="#">...</a>
    <ul class="nav level-2">
      <li>
        <a href="#">...</a>
      </li>
      <li class="active">
        <a href="#">...</a>
      </li>
      ...
    </ul>
  </li>
</ul>
```

_site-navigation.scss

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

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

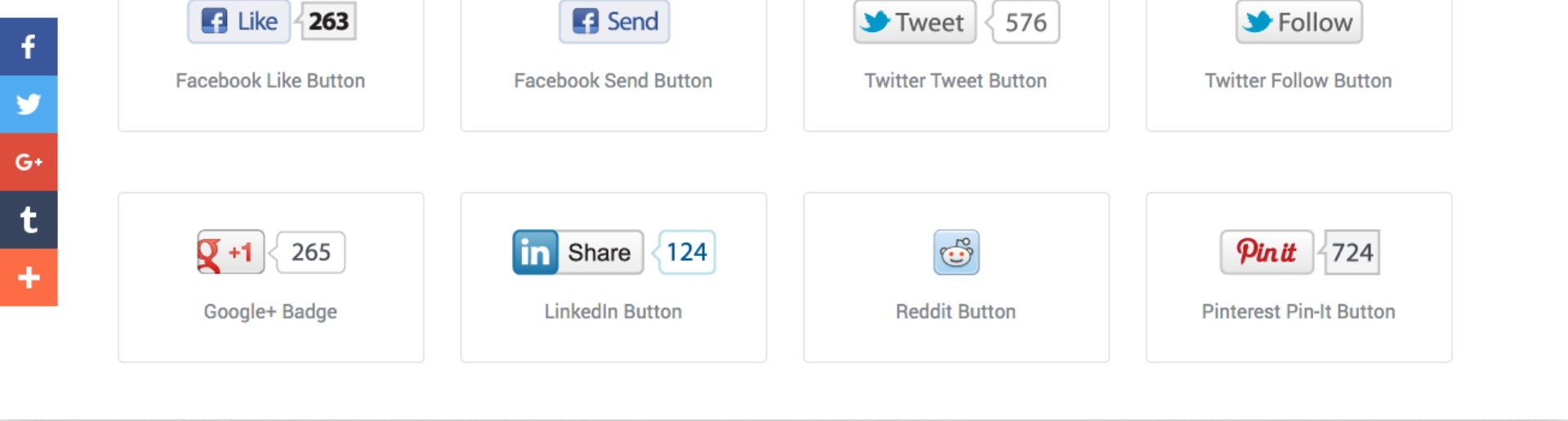
SCORE Social

SCORE's social sharing component

SCORE Social

AddThis

- Add This Plugin



09

SCORE 2.0 Migration

Migration of existing sites or starting fresh with 2.0 UI

SCORE 2.0 Migration

3 Scenerios:

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

Thank you

Ben Cato
ben@brainjocks.com

Emily Lord
emily@brainjocks.com

BRAINJOCKS
SCORE™