Description: Class notes from Jogesh K. Muppala

Associate Professor
The Hong Kong University of Science and Technology

Getting Started with Bootstrap

Objectives and Outcomes

This exercise introduces the first set of steps to set up your web page to make use of Bootstrap classes and components. At the end of this exercise, you will be able to:

- Download Bootstrap using NPM and include it in your project
- Understand how to set up a web project to use Bootstrap
- Include the Bootstrap CSS and JS classes into a web page

Note: Please remember to retain the folder and all the files that you create in this exercise. Further exercises will build upon the files that you create in this exercise. DO NOT DELETE the files at the end of the exercise.

Setting up the Project Folder

- Go to a convenient folder location on your computer and download the *Bootstrap4-starter.zip* file using the link provided at the top of this page.
- Unzip the file to see a folder named *Bootstrap4* and a sub-folder under it named *conFusion* created. Move to the *conFusion*folder.
- Open a cmd window/terminal and move to the conFusion folder.
- At the prompt type


```
npm install
```

- This will install the lite-server node module to your project.

- Next, initialize a Git repository in the project folder, and then set up a .gitignore file with the contents as shown below:

```
node_modules
```

- Now do a commit of your project folder to the Git repository with the message "Initial Setup". You will be doing a commit of your project at the end of each exercise so that you retain the completed files of each exercise.
- Set up an online Git repository and synchronize your project folder with the online repository.

Downloading Bootstrap

- You will use npm to fetch the Bootstrap files for use within your project. Thereafter you need to install JQuery and Popper.js as shown below since Bootstrap 4 depends on these two. At the prompt, type the following to fetch Bootstrap files to your project folder:

```
npm install bootstrap@4.0.0 --save
npm install jquery@3.3.1 popper.js@1.12.9 --save
```

- This will fetch the Bootstrap files and store is in your node_modules folder in a bootstrap folder. The bootstrap->dist folder contains the precompiled Bootstrap CSS and JS files for use within your project.
- Open your project folder in your editor, and then open the index.html file in the *conFusion* folder. This is your starting web page for the project. We have already created the web page with some content to get you started. We will use Bootstrap to style this web page, and learn Bootstrap features, classes and components along the way.
- Start your lite-server by typing npm start at the prompt. The *index.html* file should now be loaded into your default browser.

Getting your Web page Bootstrap ready

- Open the *index.html* file in your favourite text editor. If you are using Visual Studio Code, Brackets, Sublime Text or similar editors, you can open the project folder in the editor and then view index.html.
- Insert the following code in the *<head>* of *index.html* file before the title.

```
  <!-- Required meta tags always come first -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink
```

```
    -to-fit=no">
  <meta http-equiv="x-ua-compatible" content="ie=edge">
  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="node_modules/bootstrap/dist/css/bootstrap.min
    .css">
```

- This will include Bootstrap CSS into your web page. Note the subtle change in the fonts of the content of the web page. This is the Bootstrap typography effect coming into play. The default Bootstrap typography sets the font to Helvetica Neue and selects the appropriate font size based on the choice of the heading style and paragraph style for the content.
- At the bottom of the page, just before the end of the body tag, add the following code to include the JQuery library, popper.js library and Bootstrap's Javascript plugins. Bootstrap by default uses the JQuery Javascript library for its Javascript plugins. Hence the need to include JQuery library in the web page.

```
  <!-- jQuery first, then Popper.js, then Bootstrap JS. -->
  <script src="node_modules/jquery/dist/jquery.slim.min.js"></script>
  <script src="node_modules/popper.js/dist/umd/popper.min.js"></script>
  <script src="node_modules/bootstrap/dist/js/bootstrap.min.js"></script>
```

- Now, do a Git commit with the message "Intro. to Bootstrap". You may push the commit to your online repository.

Conclusion

We have now understood how to set up a web project to use Bootstrap. In the next lecture, we will explore further on responsive design and Bootstrap's grid system.

Exercise: Responsive Design and Bootstrap Grid System Part 1

Objectives and Outcomes

This exercise introduces you to responsive design and Bootstrap support for mobile first responsive design through the use of the grid system. At the end of this exercise, you will be able to:

- Create responsive websites using the Bootstrap grid system
- Reordering content using push, pull and offset classes

Note: In this exercise we will continue to update the *index.html* file in the *conFusion* folder that we created and edited in the previous lecture.

Bootstrap Grid System and Responsive Design

Bootstrap is designed to be mobile first, meaning that the classes are designed such that we can begin by targeting mobile device screens first and then work upwards to larger screen sizes. The starting point for this is first through media queries. We have already added the support for media queries in the last lesson, where we added this line to the head:

```
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

The *viewport* meta tag ensures that the screen width is set to the device width and the content is rendered with this width in mind. This brings us to the second issue, designing the websites to be responsive to the size of the viewport. This is where the Bootstrap grid system comes to our aid. Bootstrap makes available four sizes, xs for extra small, sm for small, md for medium and lg for large screen sizes. We have already seen the basics of responsive design. In this exercise, we will employ the Bootstrap grid classes to design the websites. We would like our website to have the content stacked on extra small devices, but become horizontal within each row for smaller devices and beyond. Towards this goal, we will make use of the classes .col-*, .col-sm-*, col-md-*, and .col-lg-* for defining the layouts for the various device sizes. We can specify how many columns each piece of content will occupy within a row, all adding up to 12 or a multiple thereof.

Using a Container class

- We use the container class to keep content within a fixed width on the screen, determined by the size of the screen. The alternative is to use the container-fluid class to make the content automatically to span the full width of the screen. We will discuss further about this when we discuss the Bootstrap grid system in the next lecture. Add the container class to the first div right after the </header> in the file as follows.

```
<div class="container"> ...
```

Dividing the content into rows

- Let us now add the class *row* to the first-level inner *div* elements inside the container. This organizes the page into rows of content. In the next exercise, we will see how we can add other classes to the rows.

```
  <div class="row"> ...
```

Creating a Jumbotron

- Let us add the class jumbotron to the header class as shown below. This turns the header element into a Bootstrap component named Jumbotron. A jumbotron is used to showcase key content on a website. In this case we are using it to highlight the name of the restaurant.

```
    <header class="jumbotron"> ...
```

- In the header add a container class to the first inner div and a row class to the second inner div.

Creating a footer

- Finally, in the footer add a container class to the first inner div and a row class to the second inner div.

Applying column classes within each row

- In the header row, we will display the restaurant name and the description to occupy 6 columns, while we will leave six columns for displaying the restaurant logo in the future. Let us go into the jumbotron and define the classes for the inner divs as follows

```
        <div class="col-12 col-sm-6"> ... </div>
        <div class="col-12 col-sm"> ... </div>
```

- For the remaining three div rows that contain the content, let us define the classes for the inner divs as follows:

```
<div class="col-12 col-sm-4 col-md-3"> ... </div>
<div class="col col-sm col-md"> ... </div>
```

- For the footer, let us define the classes for the inner divs as follows:

```
<div class="col-4 col-sm-2"> ... </div>
<div class="col-7 col-sm-5"> ... </div>
<div class="col-12 col-sm-4"> ... </div>
<div class="col-auto"> ... </div>
```

Now you can see how the web page has been turned into a mobile-first responsive design layout.

Using Order and Offset with column layout classes

- In the content rows, we would like to have the title and description to alternate so that it gives an interesting look to the web page. For extra small screens, the default stacked layout works best. This can be accomplished by using the .order-sm-last and .order-sm-first for the first and the third rows as follows:

```
<div class="col-12 col-sm-4 order-sm-last col-md-3"> ... </div>
<div class="col col-sm order-sm-first col-md"> ... </div>
```

- For the div containing the <ul> with the site links, update the class as follows:

```
<div class="col-4 offset-1 col-sm-2">
```

- After saving all the changes, you can do a Git commit with the message "Bootstrap Grid Part 1" and push your changes to the online repository.

Conclusion

In this exercise, we reviewed responsive design and the Bootstrap grid system.

Exercise: Responsive Design and Bootstrap Grid System Part 2

Objectives and Outcomes

This exercise continues the examination of responsive design and Bootstrap support for mobile first responsive design through the use of the grid system. We also learn how to customize some of the Bootstrap classes through defining our own modifications in a separate CSS file. At the end of this exercise, you will be able to:

- Customize the CSS classes through your own additions in a separate CSS file
- Centering the content both vertically and horizontally within a row

List styles

- You can use several list styles to display lists in different formats. In this exercise, we will use the unordered list style *list-unstyled* to display the links at the bottom of the page without the bullets. To do this, go to the links in the footer and update the ul as follows

```
<ul class="list-unstyled"> ... </ul>
```

Using Custom CSS classes

We can define our own custom CSS classes in a separate CSS file, and also customize some of the built-in CSS classes. We will now attempt to do this in this part of the exercise.

- Create a folder named *css*. Then create a file named *styles.css* in the *css* folder. Open this file to edit the contents. Add the following CSS code to the file:

```
.row-header{
  margin:0px auto;
  padding:0px;
}
.row-content {
  margin:0px auto;
  padding: 50px 0px 50px 0px;
  border-bottom: 1px ridge;
```

```
  min-height:400px;
}
.footer{
  background-color: #D1C4E9;
  margin:0px auto;
  padding: 20px 0px 20px 0px;
}
```

- Include the *styles.css* file into the head of the *index.html* file as follows:

```
  <link href="css/styles.css" rel="stylesheet">
```

- Then add these classes to the corresponding rows in the *index.html* file as follows. See the difference in the *index.html* file in the browser. The first one is for the row in the <header>, the next three for the rows in the content, and the last one directly to the <footer> tag.

```
  <div class="row row-header"> ... </div>
  <div class="row row-content"> ... </div>
  <div class="row row-content"> ... </div>
  <div class="row row-content"> ... </div>
  <footer class="footer"> ... </footer>
```

- Our next set of customization is to the jumbotron and the address. Add the following to *styles.css* file:

```
.jumbotron {
  padding:70px 30px 70px 30px;
  margin:0px auto;
  background: #9575CD ;
  color:floralwhite;
}
address{
  font-size:80%;
  margin:0px;
  color:#0f0f0f;
}
```

Vertically Centering the Content

- In the content section, update all the rows as follows:

```
    <div class="row row-content align-items-center">
```

- In the footer, update the third column div that contains the social media links as follows:

```
        <div class="col-12 col-sm-4 align-self-center">
```

Horizontally Centering the Content

- Update the copyright paragraph as follows:

```
      <div class="row justify-content-center">
        <div class="col-auto">
```

- Update the inner div containing the social media links as follows:

```
            <div class="text-center">
```

- After saving all the changes, you can do a Git commit with the message "Bootstrap Grid Part 2" and push your changes to the online repository.

Conclusion

In this exercise, we continued our review of responsive design and the Bootstrap grid system. We also learnt how to customize using our own CSS classes.

Icon Fonts

Objectives and Outcomes

In this exercise, we will learn the use of icons in web page design using Font Awesome icons, and bootstrap-social icons. At the end of this exercise, you will be able to:

- Use icons within your website to represent various entities making use of the glyphicons, font-awesome icons and bootstrap-social icons

Note: Some people have pointed out that if they have AdBlocker installed, then the font icons are not showing up in their browser.

Using Icon Fonts and Other CSS classes

- One of the most popular icon font toolkit is Font Awesome. Go to its website http://fontawesome.io/ to check out more details about this icon font. You can get Font Awesome using npm by typing the following at the prompt:

```
npm install font-awesome@4.7.0 --save
```

- Another module that we install is Bootstrap Social that enables the addition of Social buttons to our site. You can find more information about it at https://lipis.github.io/bootstrap-social/. To install it using npm, type the following at the prompt:

```
npm install bootstrap-social@5.1.1 --save
```

- We now need to include the CSS files for font awesome and bootstrap-social in the index.html file. Add the following code to the head of the file after the links for importing Bootstrap CSS classes. Do the same change to aboutus.html file:

```
  <link rel="stylesheet" href="node_modules/font-awesome/css/font-awesome.min
      .css">
  <link rel="stylesheet" href="node_modules/bootstrap-social/bootstrap-social
      .css">
```

- Let us now use some font icons in our web page and decorate it. Update the navbar's ul list items as follows in index.html:

```
                  <li class="nav-item active"><a class="nav-link" href="#"
                     ><span class="fa fa-home fa-lg"></span> Home</a></li>
```

```
                <li class="nav-item"><a class="nav-link" href="./aboutus
              .html"><span class="fa fa-info fa-lg"></span> About</a></li>
                <li class="nav-item"><a class="nav-link" href="#"><span
                   class="fa fa-list fa-lg"></span> Menu</a></li>
                <li class="nav-item"><a class="nav-link" href="#"><span
                   class="fa fa-address-card fa-lg"></span> Contact</a></li>
```

- Similarly update the navbar's ul list items as follows in aboutus.html:

```
                <li class="nav-item"><a class="nav-link" href="./index.html"
                   ><span class="fa fa-home fa-lg"></span> Home</a></li>
                <li class="nav-item active"><a class="nav-link" href="#"
                   ><span class="fa fa-info fa-lg"></span> About</a></li>
                <li class="nav-item"><a class="nav-link" href="#"><span
                   class="fa fa-list fa-lg"></span> Menu</a></li>
                <li class="nav-item"><a class="nav-link" href="#"><span
                   class="fa fa-address-card fa-lg"></span> Contact</a></li>
```

- Next, in both index.html and aboutus.html, go down to the address in the footer of the page and replace the "Tel.", "Fax" and "Email" with the corresponding font awesome based icons as follows:

```
                  <i class="fa fa-phone fa-lg"></i>: +852 1234 5678<br>
                  <i class="fa fa-fax fa-lg"></i>: +852 8765 4321<br>
                  <i class="fa fa-envelope fa-lg"></i>:
                  <a href="mailto:confusion@food.net">confusion@food.net</a>
```

- Finally, let us use the bootstrap-social CSS classes to create the social buttons in the footer in both index.html and aboutus.html, by replacing the social sites' links with the following code:

```
                <div class="text-center">
                    <a class="btn btn-social-icon btn-google" href="http
                       ://google.com/+"><i class="fa fa-google-plus"></i></a>
                    <a class="btn btn-social-icon btn-facebook" href="http
                       ://www.facebook.com/profile.php?id="><i class="fa fa
                       -facebook"></i></a>
                    <a class="btn btn-social-icon btn-linkedin" href="http
                       ://www.linkedin.com/in/"><i class="fa fa-linkedin"></i
                       ></a>
                    <a class="btn btn-social-icon btn-twitter" href="http
                       ://twitter.com/"><i class="fa fa-twitter"></i></a>
                    <a class="btn btn-social-icon btn-google" href="http
                       ://youtube.com/"><i class="fa fa-youtube"></i></a>
                    <a class="btn btn-social-icon" href="mailto:"><i class
                       ="fa fa-envelope-o"></i></a>
                </div>
```

- Save all the changes and commit to your Git repository with the message "Icon Fonts".

Conclusions

We learnt about using icon fonts in a web project.

Exercise: Buttons

Objectives and Outcomes

In this exercise, we will examine user input for a website through the use of Buttons support in Bootstrap. At the end of this exercise, you will be able to:

- Create, style and activate buttons in a web page using the button classes
- Use a Button Group to group together related buttons.

Exercise Resources

contactus.html.zip

Set up for the Exercise

- Download the *contactus.html.zip* file given above, unzip it and move the *contactus.html* to the *conFusion* folder. This file is already pre-formatted with some content.
- Set up the links in the navigation bars for all the three pages, *index.html, aboutus.html* and *contactus.html* so that we can navigate from one to the other with ease.
- Also set up the links in the footer correctly to point to the appropriate pages.

Adding a Button Bar

- We are now going to add content to *contactus.html* file to learn more about buttons and button bars. Go to the div where we specify "Button group goes here", and replace it with the following code to create a button bar containing three buttons:

```
                <div class="btn-group" role="group">
                    <a role="button" class="btn btn-primary" href="tel
```

```
                    :+85212345678"><i class="fa fa-phone"></i> Call</a>
                <a role="button" class="btn btn-info"><i class="fa fa-skype"
                    ></i> Skype</a>
                <a role="button" class="btn btn-success" href="mailto
                    :confusion@food.net"><i class="fa fa-envelope-o"></i>
                    Email</a>
            </div>
```

Note how we define the button bar using the *btn-group* class, and then add the three buttons using the *<a>* tag. In this case, the three buttons are hyperlinks that cause an action and have an *href* associated with them. So we decided to use the *<a>* tag instead of the *<button>* tag. Note how the *<a>* tags have been styled using the *btn* class.

- Remember to do a Git commit with the message "Buttons"

Conclusions

We have learnt how to add buttons and button groups to a web page.

Exercise: Forms

Objectives and Outcomes

In this exercise, we will examine user input for a website through the use of Forms support in Bootstrap. At the end of this exercise, you will be able to:

- Design a form using various form elements and style the form using Bootstrap form classes

Adding a Basic Form

- We will add a simple form to the page at the location identified by "Form goes here". Add the following code to page to create a simple horizontal form with two fields:

```
            <form>
                <div class="form-group row">
                    <label for="firstname" class="col-md-2 col-form-label"
                        >First Name</label>
```

```
                        <div class="col-md-10">
                            <input type="text" class="form-control" id
                               ="firstname" name="firstname" placeholder="First
                               Name">
                        </div>
                    </div>
                    <div class="form-group row">
                        <label for="lastname" class="col-md-2 col-form-label"
                           >Last Name</label>
                        <div class="col-md-10">
                            <input type="text" class="form-control" id
                               ="lastname" name="lastname" placeholder="Last
                               Name">
                        </div>
                    </div>
                </form>
```

This creates a form with two elements in the form. Note that the class *row* in the form enables us to use the Bootstrap grid system. Hence we can style the contents using the column classes as appropriate.

- Let us add fields to seek user's telephone number and email:

```
                    <div class="form-group row">
                        <label for="telnum" class="col-12 col-md-2 col-form
                           -label">Contact Tel.</label>
                        <div class="col-5 col-md-3">
                            <input type="tel" class="form-control" id="areacode"
                               name="areacode" placeholder="Area code">
                        </div>
                        <div class="col-7 col-md-7">
                            <input type="tel" class="form-control" id="telnum"
                               name="telnum" placeholder="Tel. number">
                        </div>
                    </div>
                    <div class="form-group row">
                        <label for="emailid" class="col-md-2 col-form-label"
                           >Email</label>
                        <div class="col-md-10">
                            <input type="email" class="form-control" id
                               ="emailid" name="emailid" placeholder="Email">
                        </div>
                    </div>
```

Adding a Checkbox and Select

- We now see the addition of a checkbox and a select element to the form. Note the styling of these elements using Bootstrap classes:

```
                    <div class="form-group row">
                        <div class="col-md-6 offset-md-2">
                            <div class="form-check">
                                <input type="checkbox" class="form-check-input"
                                   name="approve" id="approve" value="">
                                <label class="form-check-label" for="approve">
                                    <strong>May we contact you?</strong>
                                </label>
                            </div>
                        </div>
                        <div class="col-md-3 offset-md-1">
                            <select class="form-control">
                                <option>Tel.</option>
                                <option>Email</option>
                            </select>
                        </div>
                    </div>
```

Adding a textarea

- Next we add a textarea for the users to submit their feedback comments as follows:

```
                    <div class="form-group row">
                        <label for="feedback" class="col-md-2 col-form-label"
                           >Your Feedback</label>
                        <div class="col-md-10">
                            <textarea class="form-control" id="feedback" name
                               ="feedback" rows="12"></textarea>
                        </div>
                    </div>
```

Adding the Submit Button

- Finally, we add the submit button to the form as follows:

```
                    <div class="form-group row">
                        <div class="offset-md-2 col-md-10">
                            <button type="submit" class="btn btn-primary">Send
                                Feedback</button>
                        </div>
                    </div>
```

Note the declaration of the type for the button to *submit*.

- Remember to do a Git commit with the message "Forms"

Conclusions

We have learnt how to add a form and style the form using Bootstrap form classes.

Displaying Content: Tables and Cards

Objectives and Outcomes

In this exercise, we will examine tables and Bootstrap classes for styling tables. We will also examine Bootstrap cards and their use for displaying content. At the end of this exercise, you will be able to:

- Create, style and present tabular data in tables in a web page using the Bootstrap table classes
- Display content in a web page using Bootstrap cards

Set up for the Exercise

- In this exercise we will be modifying the *aboutus.html* page to add a table, a card with some content and a card with a quotation.
- Let us get started by opening *aboutus.html* page in a text editor.

Bootstrap Tables

- In this part, we will add a new row of content after the Corparate Leadership row in the page. We first start by adding a row and columns to the page as follows:

```
      <div class="row row-content">
          <div class="col-12 col-sm-9">
              <h2>Facts &amp; Figures</h2>
          </div>
           <div class="col-12 col-sm-3">
          </div>
      </div>
```

- Inside the first column of this row, insert the table as follows:

```
              <div class="table-responsive">
                  <table class="table table-striped">
                      <thead class="thead-dark">
                          <tr>
                              <th> </th>
                              <th>2013</th>
                              <th>2014</th>
                              <th>2015</th>
                          </tr>
                      </thead>
                      <tbody>
                          <tr>
                              <th>Employees</th>
                              <td>15</td>
                              <td>30</td>
                              <td>40</td>
                          </tr>
                          <tr>
                              <th>Guests Served</th>
                              <td>15000</td>
                              <td>45000</td>
                              <td>100,000</td>
                          </tr>
                          <tr>
                              <th>Special Events</th>
                              <td>3</td>
                              <td>20</td>
                              <td>45</td>
                          </tr>
```

```
                            <tr>
                                <th>Annual Turnover</th>
                                <td>$251,325</td>
                                <td>$1,250,375</td>
                                <td>~$3,000,000</td>
                            </tr>
                        </tbody>
                    </table>
                </div>
```

Note the use of *table-responsive* class to create a responsive table, and the *table-striped* and *thead-inverse* classes for styling the table.

Bootstrap Cards

- Next we add a card to the second div in the first content row as follows, updating the div first by adding the classes col-12 col-sm-6 to it and then adding the card:

```
            <div class="col-12 col-sm-6">
                <div class="card">
                    <h3 class="card-header bg-primary text-white">Facts At a
                       Glance</h3>
                    <div class="card-body">
                        <dl class="row">
                            <dt class="col-6">Started</dt>
                            <dd class="col-6">3 Feb. 2013</dd>
                            <dt class="col-6">Major Stake Holder</dt>
                            <dd class="col-6">HK Fine Foods Inc.</dd>
                            <dt class="col-6">Last Year's Turnover</dt>
                            <dd class="col-6">$1,250,375</dd>
                            <dt class="col-6">Employees</dt>
                            <dd class="col-6">40</dd>
                        </dl>
                    </div>
                </div>
            </div>
```

- Next, we add a Bootstrap card and include a quotation in the card using the blockquote typography style:

```
            <div class="col-12">
                <div class="card card-body bg-light">
                    <blockquote class="blockquote">
```

```
                        <p class="mb-0">You better cut the pizza in four pieces
                           because
                            I'm not hungry enough to eat six.</p>
                        <footer class="blockquote-footer">Yogi Berra,
                            <cite title="Source Title">The Wit and Wisdom of
                               Yogi Berra,
                            P. Pepe, Diversion Books, 2014</cite>
                        </footer>
                    </blockquote>
                </div>
            </div>
```

Note the use of the *<blockquote>* tag to create a block quote in the card. We can use a *<footer>* inside the block quote to specify the attribution of the quote to its origin.

- Remember to commit the changes to your Git repository with the message "Tables and Cards"

Conclusions

In this exercise, we constructed a table and styled it with the Bootstrap table classes. Thereafter, we added two cards to the web page. We also saw the use of the description list and the block quote in the content.

Images and Media

Objectives and Outcomes

In this exercise, we will explore the Bootstrap classes to support image and media on a website. In particular, we will look at how to include images on a website, how to make use of images within a media objects. At the end of this exercise you will be able to:

- Use Bootstrap classes to include a responsive images in a website
- Use a media object to include images and description on a website

Exercise Resources

img.zip

Set up for the Exercise

- Download the img.zip file that we provide above and unzip it in the *conFusion* folder. This should create a folder named *img*there.
- We will now update the *index.html* file to include images and media objects on the web page.

Adding the Restaurant Logo

- We will now add the restaurant logo to the Jumbotron. In index.html go to the header row inside the jumbotron and replace the second column with the following code:

```
                <div class="col-12 col-sm align-self-center">
                    <img src="img/logo.png" class="img-fluid">
                </div>
```

You will immediately notice the restaurant logo being displayed in the jumbotron.

- Next, we will add the logo to the navbar where we display the restaurant brand. Go to the navbar and replace the code there for the <a> tag with the "navbar-brand" class with the following code:

```
            <a class="navbar-brand mr-auto" href="#"><img src="img/logo.png"
                height="30" width="41"></a>
```

Note the inclusion of the logo in the navbar.

- Repeat the above two steps for the *aboutus.html* and the *contactus.html* page also to update their navbars and jumbotrons.

Adding Media Objects

- Next we will work with the content on the web page and use the media object classes to style the content in the content rows.
- Go to the first content row, and replace the content in the second column containing the description of Uthappizza with the following code:

```
                <div class="media">
                    <img class="d-flex mr-3 img-thumbnail align-self-center"
                            src="img/uthappizza.png" alt="Uthappizza">
                    <div class="media-body">
                        <h2 class="mt-0">Uthappizza</h2>
                        <p class="d-none d-sm-block">A unique combination of
                           Indian Uthappam (pancake) and
                            Italian pizza, topped with Cerignola olives, ripe
                               vine
                            cherry tomatoes, Vidalia onion, Guntur chillies and
                            Buffalo Paneer.</p>
                    </div>
                </div>
```

Note the use of the *media* class and the related Bootstrap classes to style the content.

- Next, we will go to the third row and replace the contents of the second column containing the description about Alberto with the following content:

```
                <div class="media">
                    <img class="d-flex mr-3 img-thumbnail align-self-center"
                            src="img/alberto.png" alt="Alberto Somayya">
                    <div class="media-body">
                        <h2 class="mt-0">Alberto Somayya</h2>
                        <h4>Executive Chef</h4>
                        <p class="d-none d-sm-block">Award winning three-star
                           Michelin chef with wide
                            International experience having worked closely with
                            whos-who in the culinary world, he specializes in
                            creating mouthwatering Indo-Italian fusion
                               experiences.
                            </p>
                    </div>
                </div>
```

- Finally, do a Git commit with a message "Images and Media".

Conclusions

In this exercise, we learnt about the Bootstrap classes to support images and media in a web page. We saw how we can include responsive images on a web page. In addition, we saw the use of images within a media object to style and display content.

Alerting Users

Objectives and Outcomes

In this short exercise we will examine the use of badges as a way of alerting users. At the end of this exercise, you will be able to:

- Add a badge to your web page using the Bootstrap badge class

Adding Badges

- We will continue to edit the *index.html* file. In this file, we will add a badge *HOT* next to the name of the dish Uthappizza in the first content row. To do this, add the following code inside the <*h2*> containing the name of the dish:

```
<span class="badge badge-danger">HOT</span>
```

- Next we will add a badge as a badge-pill right next to the earlier tag in the web page. Add the following code to the <*h2*> tag:

```
<span class="badge badge-pill badge-secondary">$4.99</span>
```

- Remember to commit the changes to the Git repository with message "Alerting Users".

Conclusions

In this short exercise, we learnt how to add badges to our web page.

Tabs

Objectives and Outcomes

In this exercise we will explore Bootstrap tabs and tabbed navigation. In particular we will learn about the use of tabs for organizing the content. At the end of this exercise you will be able to:

- Design a web page to use tabbed navigation for organizing the content
- Use tab panes and organize the content into the panes
- Facilitate navigation among the tab panes using the tabbed navigation elements

Adding Tab Navigation Elements

- Open the *aboutus.html* page and move to the second content row containing the details of the corporate leadership of the restaurant.
- Right after the Corporate Leadership heading, introduce the following code to set up the tabbed navigation:

```
            <ul class="nav nav-tabs">
                <li class="nav-item">
                  <a class="nav-link active" href="#peter"
                    role="tab" data-toggle="tab">Peter Pan, CEO</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="#danny" role="tab"
                    data-toggle="tab">Danny Witherspoon, CFO</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="#agumbe"role="tab"
                    data-toggle="tab">Agumbe Tang, CTO</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="#alberto" role="tab"
                    data-toggle="tab">Alberto Somayya, Exec. Chef</a>
                </li>
            </ul>
```

Note the use of the *<ul>* tag with the *nav* and *nav-tabs* classes to set up the tab navigation. Each list item within the list acts as the tab element. Within each list item, note that we set up the *<a>* tags with the *href* pointing to the *id* of the tab pane of content to be introduced later. Also note that the *<a>* tag contains the *data-toggle=tab* attribute. The first list element's *<a>* tag contains the class *active*. This tab will be the open tab when we view the web page. We can switch to the other tabs using the tabbed navigation that we just set up.

Adding Tab Content

- The details about the various corporate leaders should now be organized into various tab panes. To begin this, we will enclose the entire content into a div element with the class tab-content as specified below:

```
<div class="tab-content">
    ...

</div>
```

- Then we take the name and description of the CEO of the company and enclose it within a tab-pane as follows

```
<div role="tabpanel" class="tab-pane fade show active" id
    ="peter">
     <h3>Peter Pan <small>Chief Epicurious Officer</small
        ></h3>
     <p> ... </p>
</div>
```

Note the use of the *tab-pane, fade, show,* and *active* classes and with *peter* as the id. This is the same id used as the *href* in the *<a>* link in the navigation.

- The remaining content is also similarly enclosed inside appropriate divs with the correct ids and the classes specified as above. Only the first tab pane will have the *show* and *active* classes specified to indicate that the content should be visible on the web page by default.

Modifying the tab-content CSS

- We now modify the CSS styles for the tab-content class in the *mystyles.css* file as follows:

```
.tab-content {
   border-left: 1px solid #ddd;
   border-right: 1px solid #ddd;
   border-bottom: 1px solid #ddd;
   padding: 10px;
}
```

This modification adds a 1px border to the tab content which joins with the upper border introduced by the tab navigation element to give a clean tab like appearance.

- Finally do a Git commit with the message "Tabs".

Conclusions

In this exercise we learnt the use of tabbed navigation, tab content and tab panes and their use in organizing and navigating within the content in a page.

Accordion

Objectives and Outcomes

In this exercise we explore the use of the collapse Javascript plugin together with card component to create an accordion to show/hide content in a web page. At the end of this exercise, you will be able to:

- Design an accordion using the collapse plugin together with the card component.

Converting Tabs to Accordion

- First delete the <ul> class that was introduced for the tabbed navigation.
- Then the turn the *tab-content* div into a *accordion* div. Use the code structure as shown below:

```
<div id="accordion">
    . . .
</div>
```

- Then, convert the first tab-pane into a card such that the name appears as a card heading, and the <p> will be in the card body. Use the structure of the code as shown below:

```
<div class="card">
    <div class="card-header" role="tab" id="peterhead">
    <h3 class="mb-0">
        <a data-toggle="collapse" data-target="#peter">
        Peter Pan <small>Chief Epicurious Officer</small>
        </a>
    </h3>
    </div>
    <div class="collapse show" id="peter" data-parent
      ="#accordion">
```

```
                        <div class="card-body">
                            <p class="d-none d-sm-block">. . .</p>
                        </div>
                    </div>
                </div>
```

- For the remaining three leaders, use the same structure as above, with the appropriate ids set up for the cards, as shown in the code structure below:

```
                <div class="card">
                        <div class="card-header" role="tab" id="dannyhead">
                        <h3 class="mb-0">
                            <a class="collapsed" data-toggle="collapse" data
                               -target="#danny">
                            Dhanasekaran Witherspoon <small>Chief Food
                               Officer</small>
                            </a>
                        </h3>
                    </div>
                    <div class="collapse" id="danny" data-parent
                       ="#accordion">
                        <div class="card-body">
                            <p class="d-none d-sm-block">. . .</em></p>
                        </div>
                    </div>
                </div>
                <div class="card">
                        <div class="card-header" role="tab" id="agumbehead">
                        <h3 class="mb-0">
                            <a class="collapsed" data-toggle="collapse" data
                               -target="#agumbe">
                            Agumbe Tang <small>Chief Taste Officer</small>
                            </a>
                        </h3>
                    </div>
                    <div class="collapse" id="agumbe" data-parent
                       ="#accordion">
                        <div class="card-body">
                            <p class="d-none d-sm-block">. . .</p>
                        </div>
                    </div>
                </div>
                <div class="card">
                        <div class="card-header" role="tab" id="albertohead"
                           >
                        <h3 class="mb-0">
                            <a class="collapsed" data-toggle="collapse" data
                               -target="#alberto">
```

```
                                Alberto Somayya <small>Executive Chef</small>
                                </a>
                            </h3>
                        </div>
                        <div class="collapse" id="alberto" data-parent
                            ="#accordion">
                            <div class="card-body">
                                <p class="d-none d-sm-block">. . .</em></p>
                            </div>
                        </div>
                    </div>
```

- After completing the update, check the behavior of the accordion on the web page.
- Finally do a Git commit with the message "Accordion".

Conclusions

In this exercise we constructed the accordion using the collapse plugin together with the card component.

Tooltips and Modals

Objectives and Outcomes

In this exercise we will examine how to add tooltips to a web page. In addition we look at adding modals to a web page. At the end of this exercise, you will be able to:

- Add tooltips to a web page
- Add modals that are revealed when the user clicks on a link or a button in the web page.

Adding a Tooltip

- Let us now switch to the *index.html* page. We will now add a tooltip to this page. The tooltip will be added to the "Reserve Table" button that is in the jumbotron. We will update the *<a>* tag for the button as follows:

```
                    <a role="button" class="btn btn-block nav-link btn-warning"
                        data-toggle="tooltip" data-html="true"  title="Or Call
                            us at  <br><strong>+852 12345678</strong>"
                        data-placement="bottom" href="#reserveform">Reserve
                            Table</a>
```

As you can see from the code, we add a *data-toggle*, *data-placement* and a *title* attribute to the <a> tag in order to introduce a tooltip.

- The tooltip needs to be activated by adding a small Javascript code to the bottom of the page as follows:

```
<script>
    $(document).ready(function(){
        $('[data-toggle="tooltip"]').tooltip();
    });
</script>
```

This script is added right after the line that imports the bootstrap.min.js file.

Adding a Modal

- In the next step we introduce the modal to the web page. To set up the modal, add the following code right after the navbar at the top of the page.

```
<div id="loginModal" class="modal fade" role="dialog">
    <div class="modal-dialog modal-lg" role="content">
        <!-- Modal content-->
        <div class="modal-content">
            <div class="modal-header">
                <h4 class="modal-title">Login </h4>
                <button type="button" class="close" data-dismiss="modal"
                    >&times;</button>
            </div>
            <div class="modal-body">
                <form>
                    <div class="form-row">
                        <div class="form-group col-sm-4">
                            <label class="sr-only" for
                                ="exampleInputEmail3">Email address</label
                                >
                            <input type="email" class="form-control form
                                -control-sm mr-1" id="exampleInputEmail3"
                                placeholder="Enter email">
                        </div>
                        <div class="form-group col-sm-4">
                            <label class="sr-only" for
                                ="exampleInputPassword3">Password</label>
                            <input type="password" class="form-control form
                                -control-sm mr-1" id="exampleInputPassword3"
                                placeholder="Password">
```

```
                            </div>
                            <div class="col-sm-auto">
                                <div class="form-check">
                                    <input class="form-check-input" type
                                        ="checkbox">
                                    <label class="form-check-label"> Remember me
                                    </label>
                                </div>
                            </div>
                        </div>
                        <div class="form-row">
                            <button type="button" class="btn btn-secondary btn
                                -sm ml-auto" data-dismiss="modal">Cancel</button>
                            <button type="submit" class="btn btn-primary btn-sm
                                ml-1">Sign in</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
```

- Next we introduce another link on the right side of the navbar in order to trigger the display of the modal. To do this, add the following code in the navbar after the :

```
                <span class="navbar-text">
                    <a data-toggle="modal" data-target="#loginModal">
                    <span class="fa fa-sign-in"></span> Login</a>
                </span>
```

We are introducing another link to the right of the navbar using the *navbar-text*. This contains a link with an *<a>* tag with the *data-toggle="modal"* and *data-target="#loginModal"* attributes.

- Save all the changes and do a Git commit with the message "Tooltip and Modal".

Conclusions

In this exercise we explored tooltips and modals as two ways of revealing content for the user upon clicking on a button or a link.

Carousel

Objectives and Outcomes

In this exercise we will examine the carousel component and add it to the web page. We will examine the configuration of the carousel and adding controls to the carousel. At the end of this exercise you will be able to:

- Use a carousel component in your web page
- Configure various aspects of the carousel
- Add controls to the carousel to manually control it

Adding a row for the carousel

- The carousel will be added to the *index.html* page. In this page, go to the top of the container div that contains the content of the page and add a new content row and an inner div spanning all the 12 columns as follows:

```
    <div class="row row-content">
        <div class="col">
         </div>
    </div>
```

Adding a Carousel

- Next, add the basic carousel div inside the content row that you just added as follows:

```
        <div id="mycarousel" class="carousel slide" data-ride="carousel">
        </div>
```

Adding Carousel Content

- Next add the content inside the carousel as follows:

```
                <div class="carousel-inner" role="listbox">
                    <div class="carousel-item active">
                        <img class="d-block img-fluid"
                            src="img/uthappizza.png" alt="Uthappizza">
                        <div class="carousel-caption d-none d-md-block">
                            <h2>Uthappizza <span class="badge badge-danger"
                               >HOT</span> <span class="badge badge-pill
                               badge-default">$4.99</span></h2>
```

```
                                . . .
                            </div>
                        </div>
                        <div class="carousel-item">
                                . . .
                        </div>
                        <div class="carousel-item">
                                . . .
                        </div>
                    </div>
```

Note that the first item has been set up partially. Fill in the remaining parts from the content rows below.

Adding CSS Classes

- Add the following CSS classes to the *styles.css* file:

```
.carousel {
   background:#512DA8;
}
.carousel-item {
 height: 300px;
}
.carousel-item img {
   position: absolute;
   top: 0;
   left: 0;
   min-height: 300px;
}
```

Adding Carousel Controls

- Next, we will add manual controls to the carousel so that we can manually move among the slides. Add the following code to the bottom after the carousel items in the div of the carousel to add slide indicators that enable us to select a specific slide:

```
                    <ol class="carousel-indicators">
                        <li data-target="#mycarousel" data-slide-to="0" class
                           ="active"></li>
                        <li data-target="#mycarousel" data-slide-to="1"></li>
                        <li data-target="#mycarousel" data-slide-to="2"></li>
                    </ol>
```

- Then, add the left and right controls to the carousel that enable us to move to the previous and next slide manually. Add this to the bottom of the carousel div:

```
                    <a class="carousel-control-prev" href="#mycarousel" role
                       ="button" data-slide="prev">
                        <span class="carousel-control-prev-icon"></span>
                    </a>
                    <a class="carousel-control-next" href="#mycarousel" role
                       ="button" data-slide="next">
                        <span class="carousel-control-next-icon"></span>
                    </a>
```

- Do a Git commit with the message "Carousel".

Conclusions

In this exercise we learnt about the carousel component and how to add it to a web page. We also learnt about introducing manual controls to the carousel.

Bootstrap and JQuery

Objectives and Outcomes

In this exercise we learn about using Bootstrap's JS component methods together with JQuery and JavaScript to write JavaScript code to control the JS component. We will use the Carousel as an example of a component that can be controlled. At the end of this exercise you will be able to:

- Use Bootstrap's JS component methods together with JQuery and Javascript
- Use JS code to control the Bootstrap JS component

Adding the Carousel Control Buttons

- We will introduce two new buttons into the carousel component that we already included in the index.html page. To add the two buttons to the carousel, add the following code to the end of the carousel:

```
                <div class="btn-group" id="carouselButton">
                    <button class="btn btn-danger btn-sm" id="carousel-pause">
                      <span class="fa fa-pause"></span>
```

```
                    </button>
                    <button class="btn btn-danger btn-sm" id="carousel-play">
                      <span class="fa fa-play"></span>
                    </button>
                </div>
```

We are adding the two buttons inside a button group with the ID carouselButtons. The two buttons contain the pause and play glyphicons to indicate their corresponding actions.

Adding CSS Class for the Buttons

- Next, we add the following CSS class to styles.css file to position the buttons at the bottom-right corner of the carousel:

```
#carouselButton {
   right:0px;
   position: absolute;
   bottom: 0px;
}
```

Adding JavaScript Code

- Finally we add the following JavaScript code to activate the buttons:

```
   <script>
        $(document).ready(function(){
            $("#mycarousel").carousel( { interval: 2000 } );
            $("#carousel-pause").click(function(){
                $("#mycarousel").carousel('pause');
            });
            $("#carousel-play").click(function(){
                $("#mycarousel").carousel('cycle');
            });
        });
   </script>
```

- Do a Git commit with the message "Bootstrap JQuery"

Conclusions

In this exercise we learnt about Bootstrap's JS component methods and how they can be used together with JQuery and JavaScript to control the behavior of a Bootstrap JS component.

More Bootstrap and JQuery

Objectives and Outcomes

In this exercise we extend the previous exercise of controlling the carousel by using more JQuery and JavaScript to write JavaScript code to control the JS component. At the end of this exercise you will be able to:

- Use Bootstrap's JS component methods together with JQuery and Javascript
- Use JS code to control the Bootstrap JS component

Modifying the Carousel Control Buttons

- We will modify the carousel control buttons in the carousel component that we already included in the index.html page. Instead of two buttons, we will use a single button that will indicate if the carousel is currently cycling or paused. Furthermore we can use the button to toggle the carousel cycling behavior:

```
                    <button class="btn btn-danger btn-sm" id="carouselButton">
                        <span id="carousel-button-icon" class="fa fa-pause"
                            ></span>
                    </button>
```

We are adding a single button inside a button group with the ID carouselButton. The buttons will show either as a pause or play button based on the current behavior of the carousel.

Modifying JavaScript Code

- Finally we modify the JavaScript code to control the behavior of the carousel and also show the appropriate button:

```
            $("#carouselButton").click(function(){
                if ($("#carouselButton").children("span").hasClass('fa-pause'))
                    {
                     $("#mycarousel").carousel('pause');
                     $("#carouselButton").children("span").removeClass('fa-pause'
                        );
                     $("#carouselButton").children("span").addClass('fa-play');
                }
                else if ($("#carouselButton").children("span").hasClass('fa
```

```
                    -play')){
                     $("#mycarousel").carousel('cycle');
                     $("#carouselButton").children("span").removeClass('fa-play'
                        );
                     $("#carouselButton").children("span").addClass('fa-pause');

                }
            });
```

- Do a Git commit with the message "More Bootstrap JQuery".

Conclusions

In this exercise we learnt more about Bootstrap's JS component methods and how they can be used together with JQuery and JavaScript to control the behavior of a Bootstrap JS component.

Less

Objectives and Outcomes

In this exercise you will learn to write Less code and then automatically transform it into the corresponding CSS code. At the end of this exercise you will be able to:

- Write Less code using many of the features of Less
- Automatically convert the Less code into CSS

Adding Less Variables

- Open the *conFusion* project in a text editor of your choice. In the css folder, create a file named *styles.less*. We will add the Less code into this file.
- Add the following Less variables into the file:

```
@lt-gray: #ddd;
@background-dark: #512DA8;
@background-light: #9575CD;
@background-pale: #D1C4E9;
// Height variables
@carousel-item-height: 300px;
```

We have just added a few color and a height variable. We will make use of these variables while defining the classes.

Less Mixins

- Next we add a mixin into the file as follows:

```
.zero-margin (@pad-up-dn: 0px, @pad-left-right: 0px) {
 margin:0px auto;
 padding: @pad-up-dn @pad-left-right;
}
```

We will make use of this to define several row classes next. Using the variables and Mixin class that we defined earlier, add the following row classes to the file:

```
.row-header{
    .zero-margin();
}
.row-content {
    .zero-margin(50px,0px);
    border-bottom: 1px ridge;
    min-height:400px;
}
.footer{
    background-color: @background-pale;
    .zero-margin(20px, 0px);
}
.jumbotron {
    .zero-margin(70px,30px);
    background: @background-light ;
    color:floralwhite;
}
address{
    font-size:80%;
    margin:0px;
    color:#0f0f0f;
}
body{
    padding:50px 0px 0px 0px;
    z-index:0;
}
.navbar-dark {
     background-color: @background-dark;
}
.tab-content {
```

```
    border-left: 1px solid @lt-gray;
    border-right: 1px solid @lt-gray;
    border-bottom: 1px solid @lt-gray;
    padding: 10px;
}
```

Note the use of the variables and the mixin with various parameters in defining the classes.

Nesting Selectors

- Next we add a carousel class to illustrate the use of nesting of classes in Less, as follows:

```
.carousel {
    background:@background-dark;
    .carousel-item {
        height: @carousel-item-height;
        img {
            position: absolute;
            top: 0;
            left: 0;
            min-height: 300px;
        }
    }
}
#carouselButton {
    right:0px;
    position: absolute;
    bottom: 0px;
}
```

Installing and using the lessc Compiler

- Now we install the node module to support the compilation of the Less file. To do this, type the following at the command prompt:

```
    npm install -g less@2.7.2
```

This will install the *less* NPM module globally so that it can be used by any project. Note: if you are executing this on a Mac or Linux machine, you may need to add "sudo" to the beginning of this command. This will make available the *lessc* compiler for us so that we can compile Less files.

- Next, go to the CSS folder on your machine and rename the *styles.css* file that you have there as *styles-old.css*. This is to save the CSS file that we have been using so far. We will be creating a new *styles.css* file by compiling the Less file.
- Next type the following at the command prompt to compile the Less file into a CSS file:

```
lessc styles.less styles.css
```

- You can now do a Git commit with the message "Less".

Conclusions

In this exercise you learnt to write Less code and then automatically generating the CSS file by compiling the Less code.

Scss

Objectives and Outcomes

In this exercise you will learn to write Scss code and then automatically transform it into the corresponding CSS code. At the end of this exercise you will be able to:

- Write Scss code using many of the features of Scss
- Automatically convert the Scss code into CSS

Adding Scss Variables

- Open the *conFusion* project in a text editor of your choice. In the css folder, create a file named *styles.scss*. We will add the Scss code into this file.
- Add the following Scss variables into the file:

```
$lt-gray: #ddd;
$background-dark: #512DA8;
$background-light: #9575CD;
$background-pale: #D1C4E9;
// Height variables
$carousel-item-height: 300px;
```

We have just added a few color and a height variable. We will make use of these variables while defining the classes.

Scss Mixins

- Next we add a mixin into the file as follows:

```
@mixin zero-margin($pad-up-dn, $pad-left-right) {
 margin:0px auto;
 padding: $pad-up-dn $pad-left-right;
}
```

We will make use of this to define several row classes next.

- Using the variables and Mixin class that we defined earlier, add the following row classes to the file:

```
.row-header{
    @include zero-margin(0px,0px);
}
.row-content {
    @include zero-margin(50px,0px);
    border-bottom: 1px ridge;
    min-height:400px;
}
.footer{
    background-color: $background-pale;
    @include zero-margin(20px, 0px);
}
.jumbotron {
    @include zero-margin(70px,30px);
    background: $background-light ;
    color:floralwhite;
}
address{
    font-size:80%;
    margin:0px;
    color:#0f0f0f;
}
body{
    padding:50px 0px 0px 0px;
```

```
    z-index:0;
}
.navbar-dark {
     background-color: $background-dark;
}
.tab-content {
    border-left: 1px solid $lt-gray;
    border-right: 1px solid $lt-gray;
    border-bottom: 1px solid $lt-gray;
    padding: 10px;
}
```

Note the use of the variables and the mixin with various parameters in defining the classes.

Nesting Selectors

- Next we add a carousel class to illustrate the use of nesting of classes in Scss, as follows:

```
.carousel {
    background:$background-dark;
    .carousel-item {
        height: $carousel-item-height;
        img {
            position: absolute;
            top: 0;
            left: 0;
            min-height: 300px;
        }
    }
}
#carouselButton {
    right:0px;
    position: absolute;
    bottom: 0px;
}
```

Installing and using the node-sass module

- Now we install the node module to support the compilation of the Scss file to a CSS file. To do this, type the following at the command prompt:

```
npm install --save-dev node-sass@4.7.2
```

This will install the *node-sass* NPM module into your project and also add it as a development dependency in your package.json file.

- Next open your package.json file and add the following line into the scripts object there. This adds a script to enable the compilation of the Scss file into a CSS file:

```
"scss": "node-sass -o css/ css/"
```

- In order to transform the Scss file to a CSS file, type the following at the prompt:

```
npm run scss
```

- You can now do a Git commit with the message "Sass".

Conclusions

In this exercise you learnt to write Scss code and then automatically generating the CSS file by compiling the Scss code.