Breakroom: Front-end Frameworks Integration

Due: By next class

Grading: 20 points possible

This exercise focuses on moving forward with the two main frameworks that will drive your Breakroom project; AngularJS and Twitter Bootstrap. A third part to the exercise will involve adding Font Awesome icons.

AngularJS

AngularJS is a JavaScript framework that we are using to transform Breakroom into a single page application. AngularJS accomplishes this by “extending” HTML with a series of custom attributes called *directives* that add features and functionality beyond the limits of regular HTML For this exercise, the AngularJS focus will be on loading the partial HTML files into *index.html*.

Bootstrap

Bootstrap offers a good range of front-end features for getting a responsive website off the ground quickly and we will take advantage of a number of them. For this exercise, the Bootstrap focus will be on integrating a series of its utility classes to accomplish simple but important tasks like removing list bullets, resizing images, centering elements, and more.

Font Awesome

Font Awesome offers a robust collection of font-based icons replaces what has traditionally required the creation of custom image-based icons. The beauty of Font Awesome, or any font-based icon set, is that they can be controlled by CSS.

Exercise details:

- Create local JavaScript file and add AngularJS page router code
- Update *index.html* markup
- Add Bootstrap utility classes
- Integrate Font Awesome icons

Directions

Create local js folder and js file

You will begin by “hooking” up the partial HTML files to index.html. This will require some AngularJS code that routes the appropriate partial to *index.html* depending on the which main navigation link the user clicks.

1. First, create a new local directory within your *breakroom* folder called "js".

2. Next, launch your web editor and create a new, blank, text file. Save it as "app.js" within the *js* folder you just created.

3. In *app.js*, at the top, add the following code:

   ```
   angular.module('breakroom', ['ngRoute'])
       .config(breakroomRouter);
   ```

   This establishes an AngularJS web application called "breakroom" that uses "ngRoute", a Angular module that handles routing duties. The reference to "breakroomRouter" is explained in the next step.

4. Next, add the following code below the "angular.module" code.

   ```
   function breakroomRouter($routeProvider) {
       $routeProvider
         .when('/', {
           templateUrl: 'partials/home.html'
         })
           .when('/about', {
             templateUrl: 'partials/about.html'
           })
           .when('/services', {
             templateUrl: 'partials/services.html'
           })
           .when('/portfolio', {
             templateUrl: 'partials/portfolio.html'
           })
           .when('/blog', {
             templateUrl: 'partials/blog.html'
           })
           .when('/contact', {
             templateUrl: 'partials/contact.html'
   ```

```
        })
}
```

This defines a function called "breakroomRouter" that provides the details for which partial gets loaded when a given main navigation link is clicked.

5. Finally, add the following code below the "breakroomRouter" code:

```
function navController($scope, $location) {
    $scope.isActive = function (viewLocation) {
            return viewLocation === $location.path();
    };
}
```

This defines a function called "navController" that will link the main navigation links in *index.html* to the router code in *app.js*.

Update index.html markup

6. Open *index.html*.

7. At the bottom of the document, below the link to the *angular-route.min.js* file, add the following link to your local *app.js* file.

```
    <script src="js/app.js"></script>
  </body>
</html>
```

Add AngularJS directives to index file

8. At the top of *index.html* add the following AngularJS attribute to the *html* tag.

```
<html lang="en" ng-app="breakroom">
```

This binds *index.html* to the "angular.module" code in *app.js*.

9. Wrap the unordered list that contains the main navigation links with a div and the following AngularJS attribute.

```
<div ng-controller="navController">
  <ul>
    <li>
      <a href="#">Home</a>
    <li>
```

This binds the main navigation links to the “navController” function code in *app.js*.

10. Next, update the href attribute values for main navigation links with the following:

```
<ul>
    <li>
      <a href="#/">Home</a>
    </li>
    <li>
      <a href="#/about">About</a>
   </li>
    <li>
      <a href="#/services">Services</a>
    </li>
    <li>
      <a href="#/portfolio">Portfolio</a>
    </li>
    <li>
      <a href="#/blog">Blog</a>
    </li>
    <li>
      <a href="#/contact">Contact</a>
    </li>
    <li>
      <a href="#">Login</a>
    </li>
    <li>
      <a href="#">Sign Up</a>
    </li>
</ul>
```

The “# + / + partial name” keeps the index page loaded in the browser while passing to the AngularJS router the partial to load.

Add AngularJS ng-view directive

11. Finally, in the *main* tag add the following *ng-view* directive:

```
<main ng-view>
  <!-- AngularJS partials to be loaded here -->
</main>
```

Wherever the *ng-view* directive gets placed is where the partials will get rendered in the in the page. In our case, between the *main* tags.

Test using local server

12. Testing an AngularJS app requires a web server, ideally a local web server. If you are using Brackets, you can use it’s Live Preview feature. With Brackets, you simply click the lightning icon in the upper-right corner of the application and it will launch Chrome and load the page with a local server URL like: http://127.0.0.1:52606/breakroom/#/

    If you are not using Brackets, there are alternatives including adding an local server extension to Chrome or Firefox. If you have MAMP installed on your personal computer, that will work. Otherwise, you can upload it to your web host.

Add Bootstrap utility classes

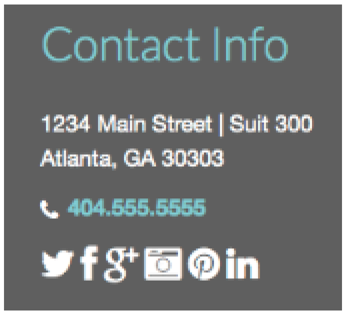
This section focuses on integrating Bootstrap utility classes in preparation for the custom CSS work you will do in the coming weeks.

13. Start by opening the Breakroom screenshots (1024 size) in Photoshop, Preview, or any application to will open PNG files. You will need to reference these to complete the following steps.

14. In *index.html*, add the “list-inline” class to the ul tag that contains the main navigation. This class will switch the vertical list to a horizontal list and remove the bullets.

```
<div ng-controller="navController">
  <ul class="list-inline">
```

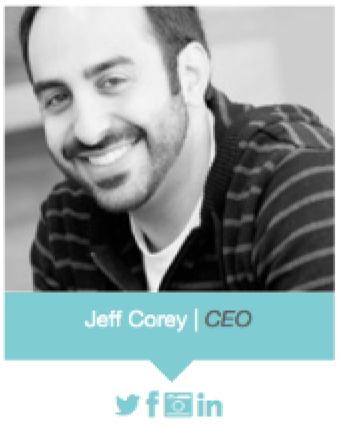
15. Repeat for social media list in *index.html* footer, and team member social media lists in *about.html*.



Contact Info

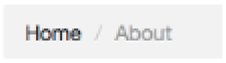
1234 Main Street | Suit 300
Atlanta, GA 30303

404.555.5555



16. In *about.html*, *services.html*, *portfolio.html*, *blog.html*, and *contact.html*, add the “breadcrumb” class to the ordered list at the top of each file. This will make the ordered list horizontal as well as include some color and padding.

```
<section>
    <ol class="breadcrumb">
```



17. In *services.html* and *blog.html*, add “list-unstyled” class where there are currently vertical lists with bullet points. Compare to screenshots to determine where to apply. This class removes the bullets but keeps the list vertical.

```
<h2>Basic</h2>
  <ul class="list-unstyled">
```

18. Add “img-responsive” class to all images. This is some advance work on the responsive grid system you will implement next class. This utility class forces an image to be sized according the to its containing element.

```
<img src="http://placehold.it/1900x520" alt="Placeholder image"
class="img-responsive">
```

Finally, you will add the “container” class throughout the Breakroom HTML files. The “container” class centers content in the browser window and restricts the width to 1170px. Later, when you implement the grid system, this width change depending on the device.

19. Start with the *header* tag in *index.html*. Within the *header* tag, wrap the header content with a new *div* tag and and give it class with value “container”.

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

20. Next, go to the first *section* tag within the *footer* tag in *index.html*. Within the section tag, wrap the content with a new *div* tag and give it class with value “container”.

21. Next, repeat same process for the second *section* tag within the *footer* tag.

```
<footer>
  <section>
    <div class="container">
      ...
    </div>
```

```
      </section>
      <section>
        <div class="container">
          ...
        </div>
      </section>
    </footer>
```

22. Next, repeat same process for all *section* tags within the partial file. That is, follow the same pattern where if you encounter a section tag, wrap the content with a new *div* tag and give it class with value “container”.

Font Awesome

For this last part of the exercise, you will integrate Font Awesome icons throughout the Breakroom files. In some cases, I’ll provide sample code, in other cases you will need to compare the screenshots with the icons listed at http://fortawesome.github.io/Font-Awesome/icons/.

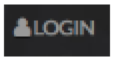
Font Awesome icons use the following syntax:

```
    <i class="fa fa-user"></i>
```

An empty *italics* tag with a class attribute containing values “fa” followed by the specific value identifying the icon, e.g. “fa-user”. There are also additional class values for controlling icon size. See examples here: http://fortawesome.github.io/Font-Awesome/examples/

23. Start by adding Font Awesome icon for user login in the header in *index.html*.

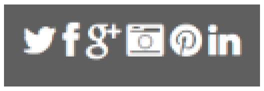
```
    <a class="login"><i class="fa fa-user"></i> Login</a>
```



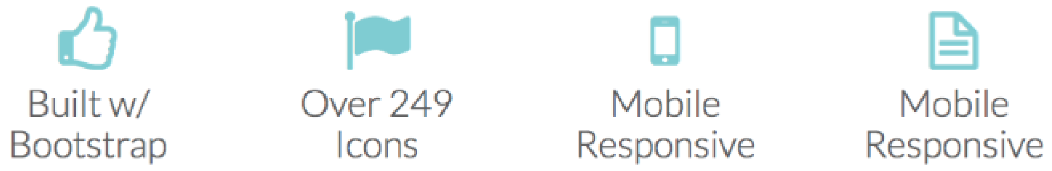
24. Next, in the footer, add social media icons, replacing the text with the icon markup for Twitter, Facebook, Google +, Intagram, Pinterest, and Linkedin.

```
    <ul class="list-inline">
      <li>
```

```
        <a href="http://www.twitter.com/themearmada">
          <i class="fa fa-twitter fa-lg"></i>
        </a>
      </li>
```

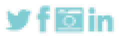


25. In *home.html*, locate and add Font Awesome icons for the following:

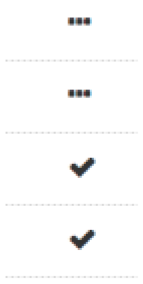


```
    <i class="fa fa-thumbs-o-up fa-4x"></i>
    <h2>Built w/ Bootstrap</h2>
```

26. In *about.html* replace text with social media icons.



27. In *services.html* replace text with ellipses and check icons.



```
    <li><i class="fa fa-ellipsis-h fa-lg"></i></li>
    <li><i class="fa fa-check fa-lg"></i></li>
```

28. Add Blog page Font Awesome icons - replace text

```
<li><a href="#"><i class="fa fa-chevron-right"></i> Theme Armada</a></li>
```

Preview your work

29. Test to see that the Bootstrap utility classes are working as well as the Font Awesome icons. If not, try checking the syntax of you class names. The slightest misspelling will prevent a class from loading.

Transfer project folder to web host

30. Use an FTP client like Fugu or Filezilla to transfer your project folder to your remote account.

Test after uploading

2. Launch a web browser, go to your project URL, and make sure that your loads as expected.

Notify instructor when exercise is complete

3. Email me the URL to your remote *breakroom* project at chris.clark@mail.ccsf.edu.

For next class

- Article: Responsive Web Design: What It Is and How To Use It (older article but the principles still hold)
- Bootstrap grid system
- Bootstrap grid system: W3Schools
- Video: Understanding the Bootstrap Responsive CSS Layout Grid (15:30 - relevant part starts 3:55 into it)