Breakroom: Integrating Bootstrap's Grid System

Due: Next class

Grading: 20 points possible

Laying out web pages can be a real pain in the ass! HTML just wasn't designed with multi-column layouts in mind. The CSS float and position properties help but unless you have a systematic approach that breaks up a design into a grid that can be implemented easily in code, the frustration quotient remains high. Oh, and the layouts need to be responsive too! Enter grid systems, and responsive ones at that. This exercise focuses on integrating Bootstrap 3's responsive grid system into Breakroom.

Grid systems

Put simply, grid systems are designed to help developers create layouts quickly, typically using a library of predefined CSS classes to create columns of varying widths. Various grid systems differ in their approach to producing layouts but, in general, they all have the following in common:

- Content area is divided into regular column units, usually 12 or 16
- Columns are float-based
- Width property establishes column widths
- Margin or padding on either side of column to create gutter space, usually 10px or 15px
- A system of class names to define column widths
- Container class to contain and center layouts
- Additional utility classes that work with grid CSS

You will be using Bootstrap 3's grid system. Bootstrap 3's grid system features the following:

- 12 column divisions
- Columns are float-based
- Column widths are percentage-based
- Gutter space is 15px of padding to the left and right side of each column
- A system of class names to define column widths
- Container class to contain and center layouts
- Grid is responsive with container class set to either 750px, 970px, or 1170px widths
- Media queries are set up to be mobile-first
- Additional utility classes (explained below)

Responsive grid systems

Grid systems aren't necessarily responsive though this is quickly becoming an expectation. Bootstrap's grid system *is* responsive, and what's more, it *mobile-first*. The fact that it's responsive means that there are *media queries* present in the Bootstrap CSS that trigger different layouts depending on the resolution of the device that is accessing it. A *mobile-first* approach means that the media queries are prioritized in a way that optimizes the CSS for mobile devices first, delivering lean CSS for smaller screens and triggering more features as the device resolutions get larger. The Bootstrap media query breakpoints are as follows:

Mobile

The Bootstrap grid delivers rules for mobile devices by default; that's device resolutions up to *768px*.

Small

Devices with resolutions starting at *768px* trigger the following media query:

```
@media (min-width: 768px){
  ... small device css here ...
}
```

Medium

Devices with resolutions starting at *992px* trigger the following media query:

```
@media (min-width: 992px){
 ... medium device css here ...
}
```

Large

Devices with resolutions starting at *1200px* trigger the following media query:

```
@media (min-width: 1200px){
 ... large device css here ...
}
```

Linking to Bootstrap CSS

Bootstrap CSS (and JavaScript) files can be downloaded and served from your local directory. Alternatively, you can link the Bootstrap CSS remotely to Bootstrap CDN (content delivery network), which is designed to deliver Bootstrap code fast. This the method you will use for this course. The style guide index file uses this method as well.

```
<link
href="http://netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css"
rel="stylesheet">
```

Viewport meta tag

In order to get mobile devices to accurately report their width, your HTML document needs to have the following present with the *head* tags. If not preset, the media queries will not work.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Column classes

Bootstrap 3 uses a system of predefined class names that give you 2 options:

- Define column width: select 1 thru 12 columns
- Define break point at which columns go from horizontal to vertical: choices are xs, sm, md, or lg

Combined together, the syntax is *.col-**[xs, sm, md, or lg]**-**[1-12]***. When defining multiple columns, the total should equal 12. For example, the markup for a row of 2 columns that go from horizontal to vertically stacked for devices with widths of less than 768px would look something like this:

```
<div class="container">
	<div class="row">
		<div class="col-sm-6">
			<h2>Heading 2</h2>
			<p>Lorem Ipsum ....</p>
		</div>
		<div class="col-sm-6">
			<h2>Heading 2</h2>
			<p>Lorem Ipsum ....</p>
		</div>
	</div>
</div>
```

If you want to maintain the 2 columns side-by-side even for the smallest resolutions, just change the class values to **col-xs-6**. There are a number of combinations you can use to create complex layouts that involve using multiple column class names. More on that further in this assignment.

Container class

You may have noticed that the previous example markup includes a parent *div* with a *container* class. The *.container div* wraps and centers layouts. Also, depending on the device resolution, the container width will differ.

```
@media (min-width: 768px) {
     .container {
          width: 750px;
     }
}

@media (min-width: 992px) {
     .container {
          width: 970px;
     }
}

@media (min-width: 1200px) {
     .container {
          width: 1170px;
     }
}
```

Row class

The *.row* class is designed to sit within *.container divs* and wrap columns, making sure that content below the row of columns are cleared of any float weirdness.

```
.row {
     margin-right: -15px;
     margin-left: -15px;
}

.row:before,
.row:after {
     display: table;
     content: " ";
}
```

```
.row:after {
     clear: both;
}

.row:before,
.row:after {
     display: table;
     content: " ";
}

.row:after {
     clear: both;
}
```

Responsive image class

The *.img-responsive* class is used to get image widths and heights sized according to the size of the containing element.

```
.img-responsive {
     display: block;
     height: auto;
     max-width: 100%;
}
```

```
<img src="path/to/image" alt="my image" class="img-responsive">
```

Other Bootstrap grid classes

The are additional useful classes related to manipulating the grid documented at getboostrap.com/css/#grid.

Bootstrap grid markup pattern

As you lay out your project pages using Bootstrap grid classes, a simple pattern to remember is "*.container div* wraps *.row div* wraps *.column divs*". For example:

```
<div class="container">
    <div class="row">
         <div class="col-sm-6">
              ... column content ...
```

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

Exercise details:

- Add viewport meta tag
- Add columns to index header & footer
- Add columns to partials
- Test Chrome mobile device emulator

Directions

1. Launch Photoshop or similar app and open the 1024px screenshots of the Breakroom design. You will need to reference these for defining column configuration.

Viewport meta tag

2. Launch an editor and open *index.html*. Add the viewport meta tag between the *head* tags:

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  ...
```

Add column classes to header

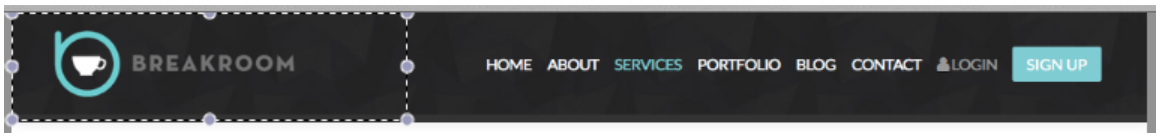
3. Start off by wrapping the content within the *.container* class with a new div and give it a class attribute with value “row”.

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

4. Within the *.row* class, wrap a new div around the linked logo.

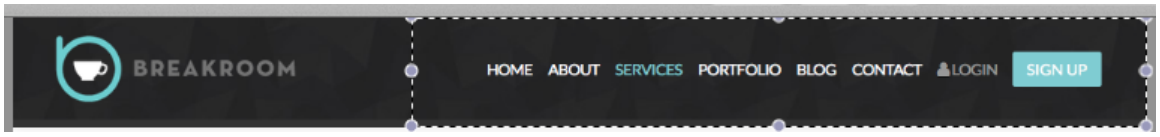
5. Add a column class to the div that spans 4 columns and targets the “sm” breakpoint, or 768px, below which the column switches to vertical stacking for smaller devices.

```
<div class="row">
  <div class="col-sm-4">
    <a href="index.html">
      <img src="images/logo.png" alt="Breakroom logo" class="img-responsive">
    </a>
  </div>
  ...
```



6. Wrap the div containing the “navController” attribute with a new div and add column class that spans 8 columns and targets the “sm” breakpoint.

```
<div class="col-sm-8">
  <div ng-controller="navController">
    <ul class="list-inline">
      ...
```



Add columns to footer

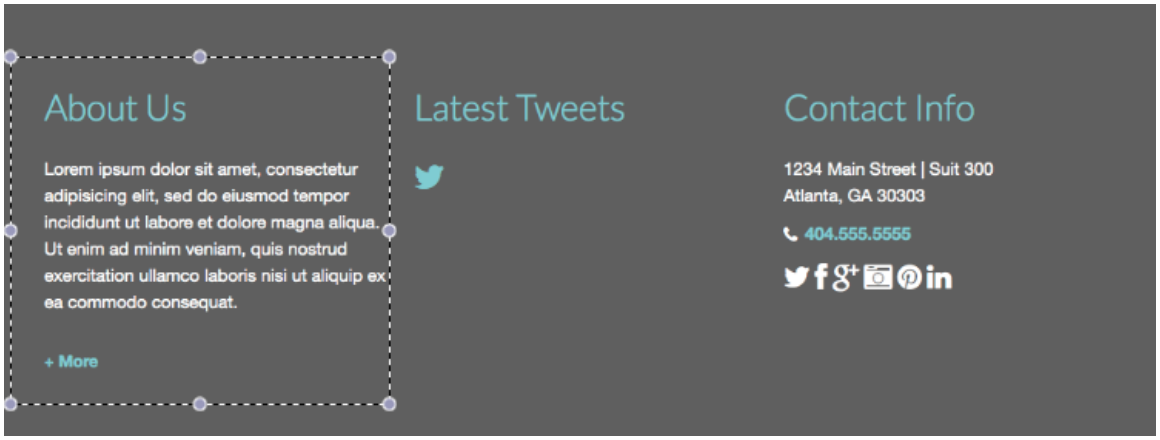
7. Add a div with .row class within the first footer section, wrapping it’s content.

```
<footer>
  <section>
    <div class="container">
      <div class="row">
```

```
<h2>About Us</h2>
...
```

8. Add a column class that spans 4 columns and targets the “sm” breakpoint.

```
<div class="row">
  <div class="col-sm-4">
    <h2>About Us</h2>
    ...
```



9. Repeat for remaining sub-sections: “Latest Tweets” and “Contact Info”.

Add columns to partials

Using the same pattern established for defining columns in the header and footer (*.container* > *.row* > *.col-x-x*), go through all of the partials and add row and column classes where appropriate. Use the screenshots for reference.

Home

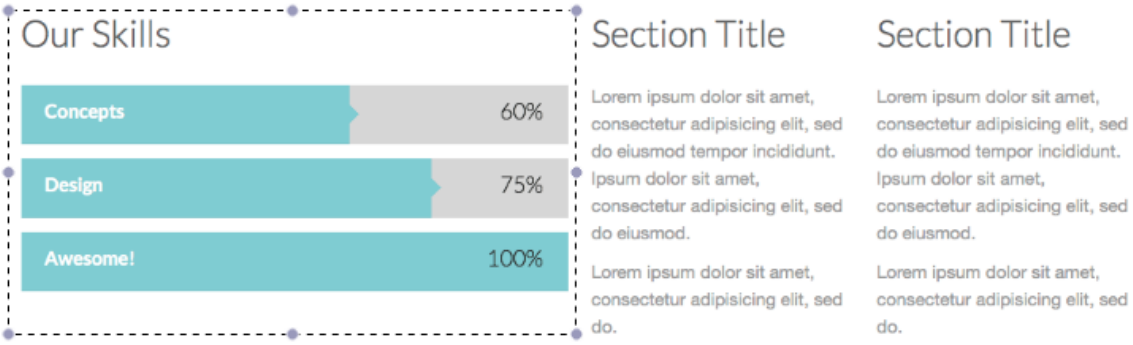
10. Add *.row* class to section.

11. Add column classes creating 4 columns. Use the “sm” breakpoint designation.

12. Add *.row* class to section.

13. Add column classes creating 6 columns. Use the “sm” breakpoint designation.

About



14. Add *.row* class to section.

15. Add column classes creating 3 columns. First column should span 6 columns. Use the “sm” breakpoint designation.

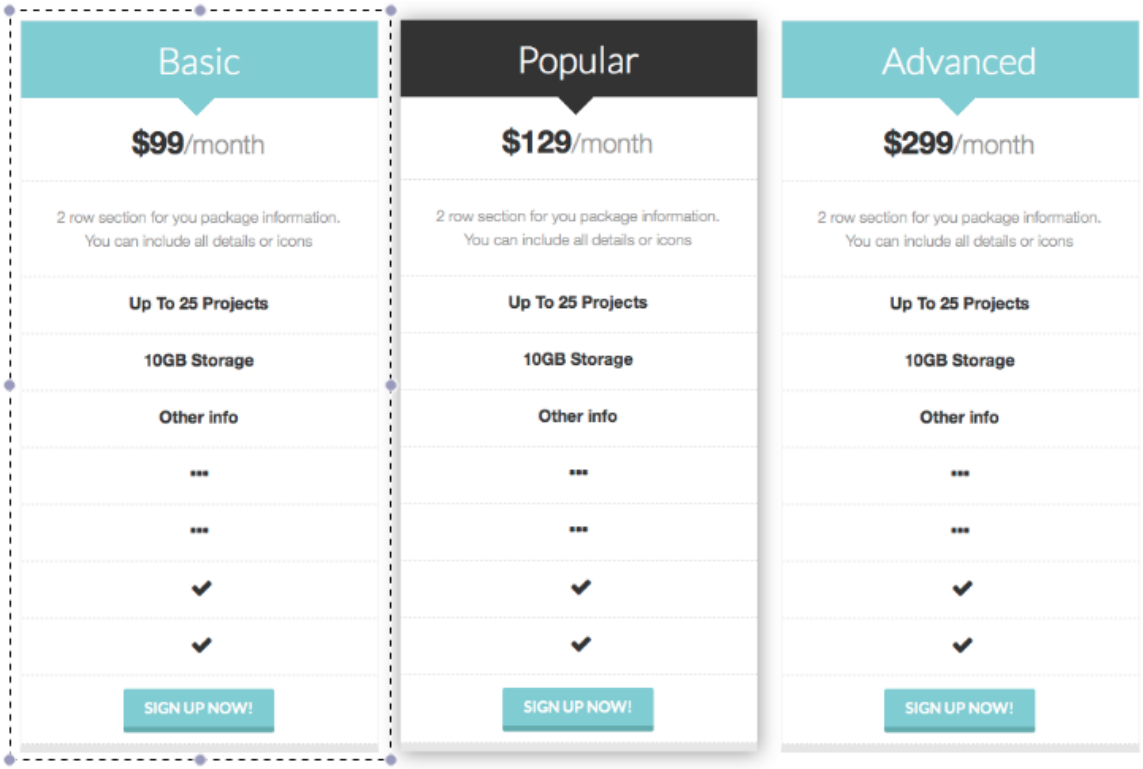
16. Add *.row* class to section.

17. Add column classes creating 4 columns. Use the “sm” breakpoint designation.

Services

18. Add *.row* class to section.

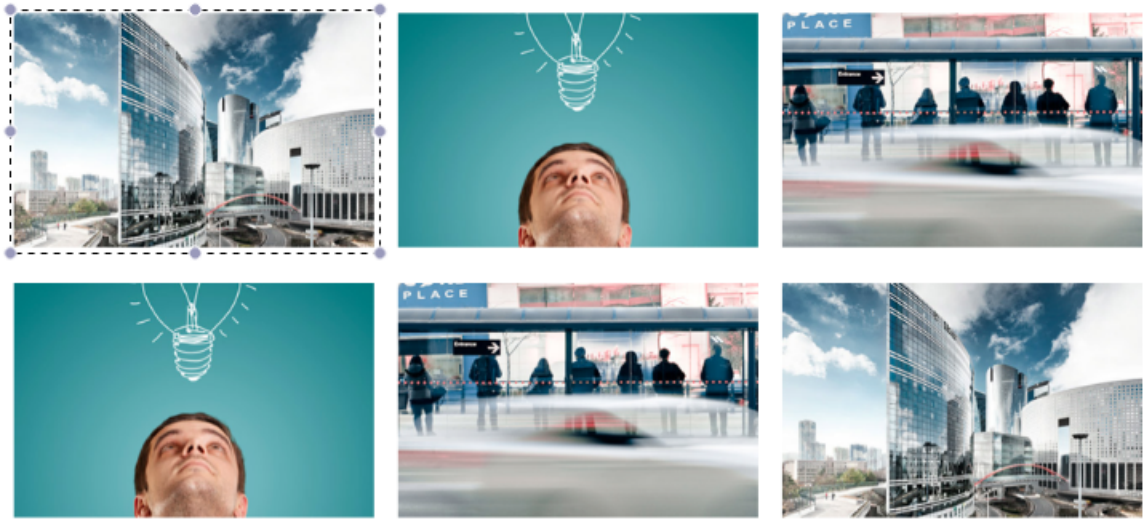
19. Add column classes creating 2 rows of 2 columns. Each column class should span 6 columns. Use the “sm” breakpoint designation.



20. Add *.row* class to section.

21. Add column classes creating 3 columns. Use the “sm” breakpoint designation.

Portfolio



22. Add *.row* class to section.

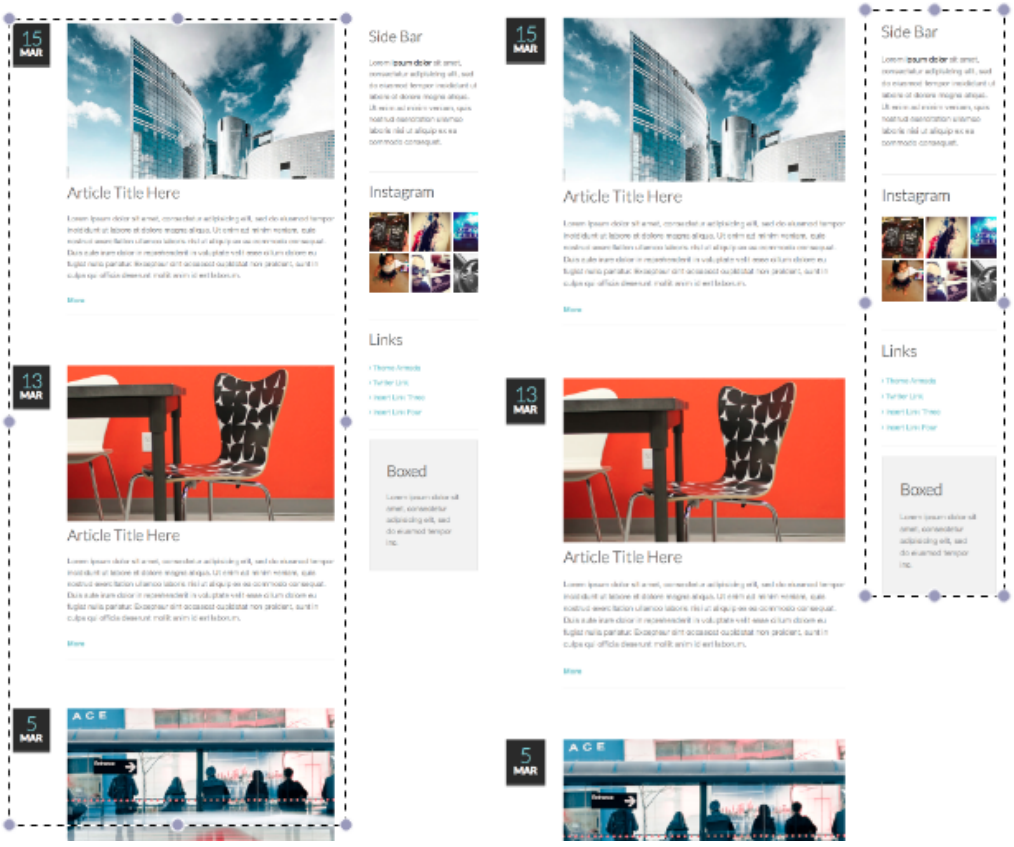
23. Add column classes creating 2 rows of 3 columns. Each column class should span 4 columns. Use the “sm” breakpoint designation.

    Note that the portfolio-related text is still visible.

24. Configure the columns for the client images on this page the same as Home.

Blog

The blog page contains 2 sets of column classes; one define 2 major columns separating the main content articles from the sidebar content. The next set of column classes get nested and separate the articles from their dates.



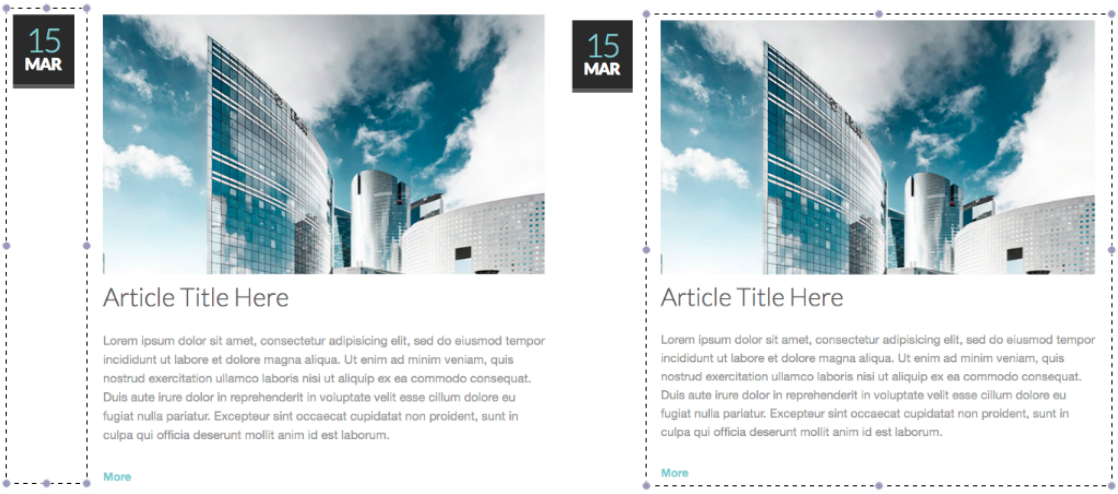
25. Add *.row* class to section. The *.row* class should span all content (articles and asides) within the *section* tags.

```
<section>
```

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

26. Add column classes creating 2 columns. First column class should span 9 columns (main content), the second 3 (sidebar). Use the “sm” breakpoint designation.

```
<section>
  <div class="container">
    <div class="row">
      <div class="col-sm-9">
        <article>
```



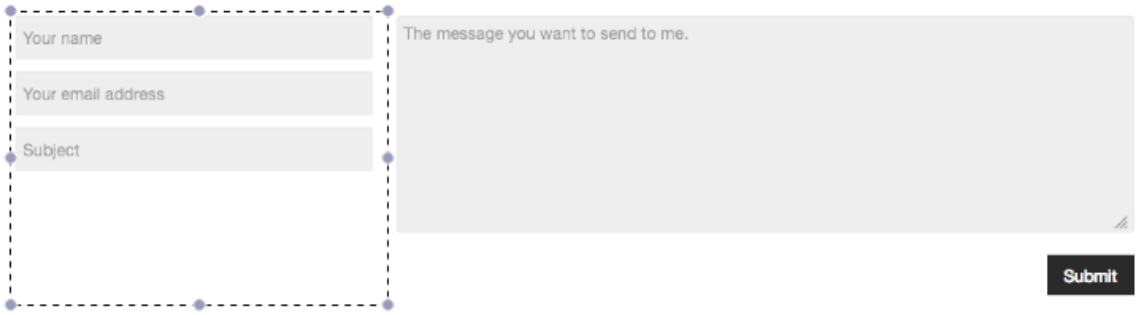

```
<div class="col-sm-3">
  <aside>
   ...
```

27. Next, within the main column containing the articles, separate the date information from the the article content with two columns; one spanning 2 columns, the other spanning 10.

```
<div class="col-sm-9">
  <article>
    <div class="col-sm-2">
      <h2>15</h2>
```

```
      <h3>MAR</h3>
    </div>
    <div class="col-sm-10">
      <img src="images/blog1.png" alt="Title" class="img-responsive">
      <h2>Article Title Here</h2>
      ...
```

Contact



28. Add *.row* class to section.

29. Add column classes creating 2 columns. Group the first 3 text fields within the form with a column class spanning 4 columns. Wrap the textarea field and submit button with column class spanning 8 columns. Use the “sm” breakpoint designation.

Test with Chrome mobile device emulator

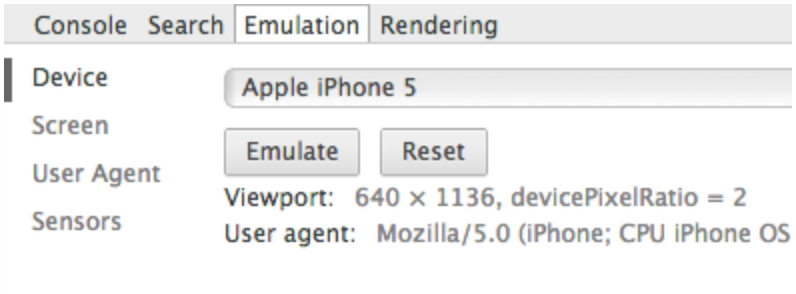
Short of testing with an array of actual devices, Chrome (and Firefox) contains a built-in mobile device emulator.

30. Launch Chrome and open *index.html*.

31. Next, open Chrome Dev Tools by either right-clicking on the page and selecting “Inspect Element” or by using the “alt-command i” keystroke combination.

32. Next, look for the “Show Console” icon  and press it.

33. Next, click the “Emulation” tab.



34. Test with the following devices, which map fairly closely with Bootstrap's breakpoints:

35.

- Apple iPhone 5
- Amazon Kindle Fire HD 7"
- Apple iPad 1/2/ iPad Mini
- Reset for high resolution desktop

Transfer project folder to web host

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

Test after uploading

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

Notify instructor when exercise is complete

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

For next class

- Pattern Tap (resource for design patterns)
- Introduction to Sass (Preprocessor for CSS)
- Scout (Utility for compiling Sass and outputting CSS)