Web Fundamentals - Session 3

Chapter 1 - What's hard about CSS?

Demo here.

You've seen quite a bit of CSS now; it all seems quite straightforward - you write some css, tweak it 'til it looks good and you're done! In theory this is exactly how CSS works and is why CSS is brilliant.

Unfortunately, the realities are not quite so straightforward. Different browsers will render CSS with subtle differences. Take a look at the cat picture. The styling is relatively simple - all we've done is add a border and a shadow.

Just because your site looks good in Chrome, doesn't mean it will look good in Internet Explorer. Making your site look good (or even presentable) in multiple browsers takes time, effort and experience.

What else is hard about CSS?

About 5 years ago, 'all' you would have had to worry about is the cross-browser display issues. Since then, the mobile web has exploded and you have another (far more important) concern: how will your site look when viewed on a mobile?

Making webpages that look good when viewed at multiple different sizes is a whole new level of complexity.

Introduction to Web Frameworks

A web application framework is a **type of software framework** designed to **support development** of dynamic websites, web applications, web services and of web resources.

A **software framework**, in computer programming, is an abstraction in which **common code** providing **generic functionality** can be selectively *overridden* or *specialized* by user code providing specific functionality.

Frameworks are a **special case** of software libraries in that they are **reusable** abstractions of code wrapped in a well-defined Application Program Interface (API), yet they contain some key distinguishing features that separate them from normal libraries.

Software frameworks have these distinguishing features that separate them from libraries or normal user applications:

1. **inversion of control** - In a framework, unlike in libraries or normal user applications, the overall program's flow of control is not dictated by the caller, but by the framework.[1]
2. **default behavior** - A framework has a default behavior. This default behavior must actually be some useful behavior and not a series of no-ops.
3. **extensibility** - A framework can be extended by the user usually by selective overriding or specialized by user code providing specific functionality.
4. **non-modifiable framework code** - The framework code, in general, is not allowed to be modified. Users can extend the framework, but not modify its code.

By using frameworks, we benefit from **peer-reviewed**, **tested** and **pre-written functionalities** that save us time and allow us to share and contribute to the wider developer community. More here.

Some Basic Development Concepts

1. An **API**, **Application Program Interface**, is an abstract description of how to use an application; it is a set of routines, protocols, and tools for building software applications. The API specifies how software components should interact and APIs are used when programming graphical user interface (GUI) components; allowing you to use code in an already functional application in a stand-alone fashion.
2. A **library** is an implementation of an API; it is a set of functions that a developer can call, usually organised into classes. It contains the compiled code that implements the functions and protocols (maintains usage state).
3. A **toolkit** is a set of libraries (APIs) and services grouped together to provide the developer with a wider range of possible solutions.
    a. For example, the Globus Toolkit provides services (such as File transferring, Job Submission and Scheduling) that a developer can install and start on their servers. They also provide API's to build applications that may use the services deployed in an integrated fashion. For example, the developer may build a program that uses the Job Submission API to communicate with the Job Submission Service.

Chapter 2 - Twitter Bootstrap

Twitter Bootstrap is a **Web Application Framework**; set of CSS (& Javascript) files, released by the makers of Twitter, and maintained by some of its developers.

Bootstrap provides a set of **ready-made CSS files** that provide pre-built **functions** for **common web development requirements**, and pre-built **solutions** to **common presentation requirements** in a **cross-browser and responsive** way.

To make use of Twitter Bootstrap, you need to do two things:

1. Link to the Twitter Bootstrap stylesheet in the head of your html page.
2. Attach the relevant Twitter Bootstrap class to your html element.

To understand how to use Bootstrap, or any framework for that matter, it is **vital to read the documentation** (it's basically a guidebook). The documentation for it is here.

Here are some basic examples for using Bootstrap, take a look.

An example: making a stripy table

Suppose you want a Zebra-esque table like this fine specimen.

Have a look at the Twitter Bootstrap table documentation and you'll discover that you need to add the table table-striped class to the <table> tag:

```
<table class="table table-striped">
  ...
</table>
```

This will apply the relevant CSS rules from the bootstrap CSS file. If you're interested, you can go into Developer Tools and view the rules that apply e.g.

```
/* from line 1950 of bootstrap.css */
.table-striped tbody tr:nth-child(odd) td,
.table-striped tbody tr:nth-child(odd) th {
        background-color: #f9f9f9;
}
```

Responsive design

Responsive design means designing your sites so that they look good on **all screen sizes**.

Twitter Bootstrap promotes a 'mobile first' philosophy, encouraging you to design your site so that it looks **good at all sizes from the very beginning.** It provides a lot of useful CSS that helps you to do this.

Bootstrap includes a **responsive**, **mobile first fluid** grid system that lets you split the screen up into 12 columns and lets you customise the size of your HTML element as a fraction of 12. See this example for a easy layout option, and look at this example for responsive design - change the size of your browser to see the difference between the "xs", "sm" and "md" classes. Can you tell what they mean?

Task:

The aim of the rest of this session will be to create the website for "Sam's Sarnies" using Twitter Bootstrap.

1. Go to the github repository for the bootstrap exercise. Download the code into your coding_course folder (by clicking 'Download ZIP' in the bottom right).
2. Open bootstrap_exercise/index.html in your browser.
3. Go to the Bootstrap website (it's hosted at github, like your first_site) and click the Download Bootstrap button (not the Download Source).
4. Unzip and copy the dist folder into the bootstrap_exercise folder.
5. Open index.html in Sublime Text and Chrome.
6. Add a link to the twitter bootstrap stylesheet into index.html

   ```
   <link href='dist/css/bootstrap.css' rel='stylesheet'>
   ```

   Note that you're using a document-relative link to a file two subfolders down.
7. Refresh the page in Chrome. Notice how the fonts have changed.
8. Add the following line to the head section:

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

   as suggested in the CSS / Overview section of the Bootstrap docs.

Chapter 3 - Bootstrap layout

Bootstrap gives you several options on how to layout your page. To find out more take a look at the Grid system docs on the Bootstrap site. Some of these options will work out-of-the-box with *responsive design* - so that you can create a single html page which will look good whether viewed on a laptop, tablet or phone.

We'll just look at a few of the most important layout options here:

Page margins

Bootstrap makes it easy to center content on your page and provide sensible page margins. To do this make use of the `container` class:

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

    <!-- page content goes here -->

  </div>
</body>
```

Columns

Bootstrap works on a grid layout, with 12 columns (by default). See a demo here. You can create a column layout with the `row` and `span` classes:

```
<div class='row'>
    <div class='col-sm-4'>
        <!-- First column content -->
    </div>
    <div class='col-sm-4'>
        <!-- Second column content -->
    </div>
    <div class='col-sm-4'>
        <!-- Third column content -->
    </div>
</div>
```

The number after the `col-sm-` determines how many of the 12 grid columns that page column takes up. The `sm` bit determines the width at which the columns will collapse on top of each other, and in this case, it represents a "small" viewport, e.g. when viewing your site on a phone. Again, for more information see the Grid system docs.

Task:

- Add a div class='container' around the page content.
- Create a row at the top of the page, with two columns, with the left twice as wide as the right. Put the h1 in the left column and the img src='images/sandwich.png' on the right.
- Create a row with three equal columns to hold each of 'The Buzz' divs.

Chapter 4 - More Bootstrap

Typography

Skim through the typography section of the bootstrap docs.

Task:

1. Change the quotes in 'The Buzz' to use blockquotes. (Don't worry about the vertical grey lines - we'll remove those later.)
2. Change the paragraphs in 'Our mission' to be lead body copy.

Badges and Buttons

Check out the buttons section (CSS > Buttons).

Task:

1. Change the 'Send' button to a success button:

```
<button class='btn btn-success'>Send</button>
```

You might also need btn-small in the Recent Activity section.

2. Make the social links at the bottom into large buttons (we'll colour them later):

```
<button class='btn btn-lg'>Send</button>
```

Images

Have a look at the image section of the Bootstrap docs (CSS > Images).

Task:

1. Make the images in 'The Buzz' round, by adding the img-circle class.
2. You can center the image by adding the alignment class text-center.
3. Change the main sandwich image into an img-responsive, as described in the responsive images section. Try resizing your browser and see how it changes size.

Chapter 5 - Modifying Bootstrap

Modifying Bootstrap

Warning: Bootstrap has been designed and heavily tested for good cross-browser compatibility. Unless you know what you're doing or have a lot of time, it's probably best to stick with their layout and tweak small things e.g. fonts and colours. Just because it looks great in your browser doesn't mean it will look great in everyone's!

If you're going to modify Bootstrap **don't touch the Bootstrap files**. Instead create a new css file of your own to overwrite anything you don't want.

This means when a new version of Bootstrap comes out you can upgrade by dragging the new version over the top of the old, without losing any modifications.

Task:

1. Create a file called main.css and write the following CSS:

```
#social-buttons button {
  color: white;
}
.btn-twitter {
  background-color: #00acee;
  border-color: #009ad5;
}
.btn-facebook {
  background-color: #4868ac;
  border-color: #314776;
}
.btn-pinterest {
  background-color: #b62f26;
  border-color: #b62f26;
}
```

2. Link this file into the head of index.html **underneath your link to bootstrap**.
3. What happens? Notice how in the first rule we've selected only those buttons that exist inside an element with id=social-buttons.

Changing the background

Task:

1. Change the background of the jumbotron to be the image fruit-and-veg.png by adding the css

```
.jumbotron {
    min-height: 600px;
    background-image: url('images/fruit-and-veg.jpg');
    background-size: cover;
    background-attachment: fixed;
}
```

2. This doesn't look quite right. The problem is that the jumbotron is it is inside the div class='container'. You can change this by moving it inside:

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


    </div>
</div>
```

   Similarly you will now need to create new containers inside the div id='buzz' and div id='mission'.

3. Change the background color of the mission section to rgba(32, 35, 41, 0.9) and the font colour to #ddd

Navbar

Browse through the navbar section of the Bootstrap docs.

Task:

1. Look at the html for the basic starter template.
2. Use it to add a navbar to your site.
3. Make it a navbar-fixed-top. You will need to add

```
body { padding-top: 70px; }
```

   to main.css.

4. Add a search box to the navbar. Use the pull-right class to put it on the right hand side.

Other things

There are various other changes you will need to make your site look like the example. Try and figure out what these are by examining the html in the Developer tools.

Task:

1. Make any other changes necessary to make your site look like the example
2. If you get stuck check by looking at the gh-pages branch on the github repo.

Homework

Finishing off

Task:

Finish off the bootstrap_exercise from class

Personal site

Task:

Continue work on your personal site. Use Twitter Bootstrap to make it **responsive**!

Further Resources on CSS

This article which explains about CSS Specificity (and more).

This article has more information on CSS selectors.

A CSS Validator