CoderDojo Drogheda – Advanced CSS

“Twitter Bootstrap”

A quick introduction

CoderDojo Drogheda - Advanced CSS

HTML Crash Course Reminders!

All pages start and end with <html> </html> tags.

The content of a page is placed between <body> and </body> tags.

A div element is like a big box which wraps all content within it.

```
<div>My Content Here</div>
```

A bullet point list is created nesting ul and li elements within one another.

```
<ul>
  <li>My first bullet point</li>
  <li>My second bullet point</li>
  <li>My third bullet point</li>
</ul>
```

Elements can also have "attributes" which give element additional properties, such as an id or class name for selecting by CSS. Example:

```
<div id="mydiv">My Dynamic Content</div>
```

Links to other pages are created using an "a" element with an attribute of "href" specifying the file to link to:

```
<a href="mypage.html">My Link</a>
```

Some other example HTML elements are:
head, title
span
table, tr, td

CSS Crash Course Reminders!

CSS is what styles our HTML

Its normal located in a CSS file and including in our html using

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

We target elements using CSS selectors:

```
# for element IDs
. for element classes
```

In the style sheet once an element is selected rules are then applied within brackets:

```
.redboldtext {
      font-weight: bold;
      color: #FF0000;
}
```

Some other example CSS rules are:

```
font-family: Arial, san-serif;
font-size: 12px;
text-align: center;
background-color: #ff0000;
background-image: url(myimage.jpg);
padding: 4px;
padding-left: 2px;
margin: 4px;
margin-right: 5px;
float: left;
float: right;
width: 100px;
height: 100px;
```

Frameworks and introducing Twitter Bootstrap

The life of a web developer can get pretty repetitive. We can spend a lot of time doing the same thing over and over again. Creating a web page, making our HTML layout, then writing the same CSS rules over and over again for each project. We use the same standard page width, the same style buttons and the same style menus. Each time, every project, spending time coding the exact same thing.... It quickly gets very boring...

A common problem shared by all programmers. Luckily, Programmers are problem solvers. If there's a problem that needs fixing, guaranteed a programmer will first think how can they build something to solve the problem. Thankfully we've solved the repetitive coding problem with a thing called "Frameworks".

A Framework is essentially a load of code that other programmers have put together to make your programming life less repetitive. Every time you start a new project, instead of spending _ages_ coding the same things you always do, with a framework, they're all ready to go for you. There are hundreds of frameworks out there, and choosing one comes down to personal preference. The one we're going to look at in this set of tutorials is the Twitter Bootstrap Framework.

NOTE!

Before you continue on, please make sure you're comfortable with HTML and CSS. The following tutorials presume you know your way around these two languages.

Getting Started / Setup

Getting up and running with the Twitter Bootstrap framework is incredibly simple. Going to their website http://twitter.github.com/bootstrap/ and click "Download Bootstrap". This will then download the entire framework in a zip file. Extract the zip file to the folder you're coding in, then open up your favourite text editor, and get started with your web page.

Lets start with a very simple HTML5 web page. Just a head with a title and a body with a h1 in it. It should look something like the text below.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Bootstrap, from Twitter</title>
  </head>
  <body>
    <h1>My Twitter Bootstrap Page</h1>
  </body>
</html>
```

You may not have come across the <!DOCTYPE html> tag at the top of the file. This just lets the web browser know that your page is a HTML5 page. As the web progressed over time, there were variations of HTML, and it's important to let the web browser know which one you were using so that it knows how to render the page, putting this at the top lets it know you want to use HTML5.

The next step is to include the CSS bootstrap file. This file contains all of the CSS rules which we would commonly use to build a web page, instead of having to rewrite all of these rules over and over again, with our framework. We just include this file in our HEAD and start using them! You should have a HTML page which looks a little like the text below.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Bootstrap, from Twitter</title>
    <link href="css/bootstrap.css" rel="stylesheet">
  </head>
  <body>
    <h1>My Twitter Bootstrap Page</h1>
  </body>
</html>
```

Basic Layout

By including the CSS file you'll notice the font of the h1 has changed! Looks a lot better but doesn't really show you what you can do with the Framework. A very common page layout has the main content centred. We could do this by making a CSS class which sets a box width and then centres it, but with Twitter Bootstrap all the CSS rules are already defined. All we have to do is add a div with the class "container"

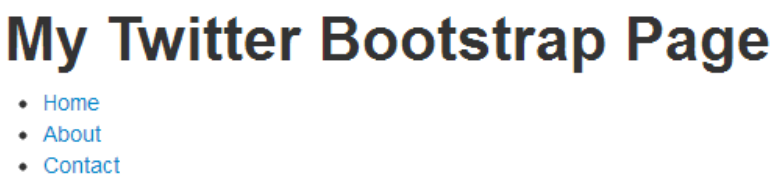
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Bootstrap, from Twitter</title>
    <link href="css/bootstrap.css" rel="stylesheet">
  </head>
  <body>
    <div class="container">
        <h1>My Twitter Bootstrap Page</h1>
    </div>
  </body>
</html>
```

The CSS rules for .container are already defined in the bootstrap.css file, ready for us to use. This is a very basic usage of the framework, next we'll take a look at a more powerful example.

Navigation

To add a menu bar to our web page is super simple. Start by adding a regular list with items linking to other pages in our web page which we'll make later. The home page list item will have a class 'active' which will indicate the page we're currently on.

```
<ul>
    <li class="active"><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
</ul>
```



Usually to start styling this list into a menu which shows all items inline, in a block with hover events would take a long time. With Twitter Bootstrap all of our rules are there, all we need to do is wrap the list in two divs with the class 'navbar' on the outter, and 'navbar-inner' on the inside div. This will give us a menu which looks like this:


```
<div class="navbar">
    <div class="navbar-inner">
        <ul class="nav">
            <li class="active"><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </div>
</div>
```

Rolling over the menu you'll see the font colour change. Our 'Home' button is also pressed in, this is the 'active' class. For future pages, change the 'active' class to be on the relevant list item.

Grid System

Now that we have a page heading and a fancy menu in place, in the centre of our page, its times to add some content. For this, we're going to introduce the concept of a "grid system".

With a grid system, the page can be easily split up into rows and columns allowing you to easily find the right position for any content on your page. With Twitter Bootstrap, your page can be split up into 12 columns, in the example below, you can see on the first row has 12 blocks, each block spans 1 column in the grid. The second row shows three blocks where each block spans 4 columns of the grid.

As you can see from the third row, you can use any combinations of blocks in order to make up your row. Using this system is incredibly simple. First we add a DIV which defines our row, then each DIV inside this, will be our blocks. If we split the page up into 3 blocks, just like the second line above, our HTML will look like this:

```
<div class="row">
    <div class="span4"></div>
    <div class="span4"></div>
    <div class="span4"></div>
</div>
```

The outer DIV builds the row simply by having the class "row" which is define in the bootstrap.css file. Each DIV inside the row builds our blocks. To build the example of 3 blocks that each span 4 columns in the grid, we add the class "span4" to each block. "span4" is a class which is defined in bootstrap.css. As you might have guessed, if you wanted the block to span any other amount of columns, you would replace the number 4 in the class name with the number of columns you want the block to span!

Content

By using the grid system it becomes really easy to add content to our web page. Adding 3 blocks spanning 4 columns each (like in the example above), we can use these to hold a few headings and some text, and we'll very quickly end up with a page that looks like this:


```
        <div class="row">
            <div class="span4">
              <h2>Heading</h2>
              <p>Donec id elit non mi…</p>
              <p><a class="btn" href="#">View details »</a></p>
            </div>
            <div class="span4">
              <h2>Heading</h2>
              <p>Donec id elit non mi porta…</p>
              <p><a class="btn" href="#">View details »</a></p>
           </div>
            <div class="span4">
              <h2>Heading</h2>
              <p>Donec sed odio dui….p>
              <p><a class="btn" href="#">View details »</a></p>
            </div>
        </div>
```

LIKE THE BUTTONS?

Another great advantage of the Twitter Bootstrap, just add the class "btn" to any link and they'll transform into a button like this. All the CSS rules are in bootstrap.css

KEEP IN MIND!

We've gotten here without writing a single line of CSS!!

A little more content

We've gotten very far without adding any new CSS rules and we have a nice menu, a nice section layout, but let's add something a bit more eye catching on our home page.

We start by adding a DIV, give it a class of "awesome". Inside the DIV add a H1, a P with the class "lead" and a link to "Read More". With the link add the classes "btn", "btn-large" and "btn-success".

```
<div class="awesome">
    <h1>Awesome Heading!</h1>
    <p class="lead">This is my super website, read on for more!!</p>
    <a class="btn btn-large btn-success" href="#">Read More</a>
</div>
```

This will produce something which looks like the image below, not quite as eye catching as what we're looking for, but almost! Twitter Bootstrap has given us the big green button, the P text is an off black colour... Almost there, but not quite.

To make it more eye catching, all we have to do is add some simple CSS rules! We'll centre all the text in .awesome then add some margins and increase the font size of the H1!

```
.awesome {
    text-align: center;
    margin-bottom: 35px;
}
.awesome h1 {
    font-size: 72px;
    margin-top: 35px;
    margin-bottom: 35px;
}
```