Bootstrap Demo

- Index page
- DFTI
- MAD

Bootstrap

Introduction

Bootstrap is a framework for cross-device user interface development. It was developed by Twitter and allows you to relatively easily create sites which are usable on both desktop and mobile devices. It uses a range of inbuilt CSS classes to allow you to create complex layouts with straightforward code.

Bootstrap development

A basic example

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Bootstrap Test</title>
<link rel="stylesheet" type="text/css"
href="http://www.free-map.org.uk/course/wad/bootstrap-3.3.5-dist/css/bootstrap.min.css" />

</head>
<body>
<h1>Hello World</h1>

</body>
</html>
```

This is an absolute minimum Bootstrap page. It does not do a lot other than display "Hello World" as an H1 heading but it demonstrates a couple of things:

- The line

  ```
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  ```

  is to ensure the screen is set up correctly for mobile devices.
- We link in Bootstrap's CSS.

The Bootstrap grid system

Bootstrap is based on a ***grid*** system (see the documentation for full details). The grid is set to have 12 columns. When creating the grid, you use s rather than tables and table rows. You use one for each row, and then a series of s withn that row for each column. By giving each a particular CSS ***class***, you can make it span a certain number of columns. Furthermore you can make a span a different number of columns on desktops and mobile devices.

An example

Firstly notice that the grid is placed inside a with a class of ***container***. ***container*** s are set up to have a "responsive fixed width" layout (i.e. a layout with fixed width but which is usable on different devices; see the Bootstrap documentation). As an alternative you can use a with a class of ***container-fluid*** which will stretch to the window width (ref: the Bootstrap documentation). You should choose one or the other for your layouts to appear correctly.

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Bootstrap Test</title>
<link rel="stylesheet" type="text/css"
href="http://www.free-map.org.uk/course/wad/bootstrap-3.3.5-dist/css/bootstrap.min.css" />
<style type="text/css">
.row div { border: 1px solid black;}
</style>
</head>
<body>
<div class="container">
<h1>Grids 1</h1>

<div class="row">
    <div class="col-md-4">
    Spans 4 columns
    </div>
     <div class="col-md-8">
    Spans 8 columns
```

```
    </div>
</div>
</div>
</body>
</html>
```

If you look at this example you can see that the row is defined with a with a class of ***row***. The Bootstrap CSS sets this up to be a row in the grid. Within this there are two sub- s which contain the content. Note the classes of ***col-md-4*** and ***col-md-8***. These will set the widths of each to 4 and 8 respectively, totalling 12 (which is the number of columns in the Bootstrap grid).

The result is:

Spans 4 columns
Spans 8 columns

Each row can have a different layout of columns. The next example illustrates this:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Bootstrap Test</title>
<link rel="stylesheet" type="text/css"
href="http://www.free-map.org.uk/course/wad/bootstrap-3.3.5-dist/css/bootstrap.min.css" />
<style type="text/css">
.row div { border: 1px solid black;}
</style>
</head>
<body>
<div class="container">
<h1>Grids 2</h1>

<div class="row">
    <div class="col-md-3">
    Spans 3 columns
    </div>
     <div class="col-md-9">
    Spans 9 columns
    </div>
</div>
<div class="row">
    <div class="col-md-4">
    Spans 4 columns
    </div>
    <div class="col-md-4">
    Spans 4 columns
    </div>
    <div class="col-md-4">
    Spans 4 columns
    </div>
</div>
</div>
</body>
</html>
```

The result is:

Spans 3 columns
Spans 9 columns
Spans 4 columns
Spans 4 columns
Spans 4 columns

The first row has two s of widths 3 and 9. The second row has three s of width 4.

Working with different devices

What does the ***md*** mean in the CSS classes in the previous example? It means "medium" device, i.e. a standard desktop. There are four classes of device (see the documentation):

- ***xs*** extra small (width < 768px), e.g. a phone;
- ***sm*** small (width 768-991px), e.g. a tablet;
- ***md*** medium (width 992-1199px);
- ***lg*** large (width 1200px or more), i.e. a desktop with unusually large screen width.

You can give your s multiple classes, and so setup different layouts for different classes of device. Here is an example (also see the documentation):

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Bootstrap Test</title>
<link rel="stylesheet" type="text/css"
href="http://www.free-map.org.uk/course/wad/bootstrap-3.3.5-dist/css/bootstrap.min.css" />
<style type="text/css">
```

```
.row div { border: 1px solid black;}
</style>
</head>
<body>
<div class="container">
<h1>Grids 3</h1>

<div class="row">
    <div class="col-md-3 col-xs-12">
    Side bar
    </div>
     <div class="col-md-9 col-xs-12">
    Main content area Yada yada lorem ipsum dolor sit amet adipscing elit Londinium et Eboracum Novum et Muscovium
        etc etc etc
    </div>
</div>
</div>
</body>
</html>
```

Note in this example how we have two s, a side bar area and a main content area. The side bar has a class of ***col-md-3*** meaning that on medium-resolution devices it will occupy 3 of the 12 grid columns. It also has a class of ***col-xs-12*** meaning that on extra-small devices it will occupy the ***whole*** of the width of the grid (all 12 columns).

The main content area has a class of ***col-md-9*** meaning that on medium-resolution devices it will occupy 9 of the 12 grid columns. Again, on extra-small devices, it will occupy the whole width of the grid due to also having a class of ***col-xs-12***.

This will mean that on medium-width devices, you'll see a side bar and main content area - while on extra-small devices, the main content area will appear ***under*** the "side" bar because the "side" bar has a width of 12 and therefore, due to occupying all 12 columns of the grid, will force the main content area onto the next row.

The results are:

Side bar
Main content area Yada yada lorem ipsum dolor sit amet adipscing elit Londinium et Eboracum Novum et Muscovium etc etc etc

Try resizing the window to simulate an extra-small device.

Layouts within layouts

You might want to give your page a more complex layout in which sub-grids are placed within the main grid. This can be done by adding an additional row nested within a content . This row would then contain its own content s. For example (also see the documentation):

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Bootstrap Test</title>
<link rel="stylesheet" type="text/css"
href="http://www.free-map.org.uk/course/wad/bootstrap-3.3.5-dist/css/bootstrap.min.css" />
<style type="text/css">
.row div { border: 1px solid black;}
</style>
</head>
<body>
<div class="container">
<h1>Grids 4</h1>

<div class="row">
    <div class="col-md-3">
    Spans 3 columns
    </div>
     <div class="col-md-9">

        <div class="row">
            <div class="col-md-4">
            Inner layout div 1
            </div>
            <div class="col-md-8">
            Inner layout div 2
            </div>
        </div>
    </div>
</div>
</div>
</body>
</html>
```

This will create an outer layout with two s, width 3 and width 9. Within the width-9 , we create ***another row*** and this inner row then has two sub- s, one with width 4 and one with width 8. So the width-9 will be divided into two inner s.

The result is:

Spans 3 columns

Sub-layout column 1
Sub-layout column 2

Clearing the row

Sometimes you might have a situation where one of the s contains much more content than the other two. An example is given below (also see the documentation):

```
<div class="row">
    <div class="col-xs-6 col-sm-6 col-md-3">
    Area 1 Yada yada lorem ipsum dolor sit amet Londinium et Eboracum Novum et Muscovium
    etc etc etc
    </div>
    <div class="col-xs-6 col-sm-6 col-md-3">
    Area 2
    </div>

    <div class="col-xs-6 col-sm-6 col-md-3">
    Area 3
    </div>
    <div class="col-xs-6 col-sm-6 col-md-3">
    Area 4
    </div>
</div>
```

As can be seen here, all four s have widths of 6 on small and extra-small devices but 3 on desktops. This will mean that on the smaller devices it should form two rows each with two s, and on the desktop it should form one row with four < div>s. However the larger amount of content in the first will cause a problem for the smaller devices. The second will appear as expected. However, because the bottom of the second will be above the bottom of the first, the third will appear directly below the second. In fact, even the bottom of the third will be above the bottom of the first, meaning the fourth, too, will appear below the third.

This problem is illustrated below: (resize the window to a narrow width)

Area 1 Yada yada lorem ipsum dolor sit amet Londinium et Eboracum Novum et Muscovium etc etc etc
Area 2
Area 3
Area 4

To deal with this we use the "clearfix" technique, which is used to solve this problem in float-based layouts. We add a "clearfix div", which is an invisible used simply to start a new line. This version will give the desired effect of the third and fourth< div>s appearing below the first and second in mobile layouts:

```
<div class="row">
    <div class="col-xs-6 col-sm-6 col-md-3">
    Area 1 Yada yada lorem ipsum dolor sit amet Londinium et Eboracum Novum et Muscovium
    etc etc etc
    </div>
    <div class="col-xs-6 col-sm-6 col-md-3">
    Area 2
    </div>

    <div class="clearfix visible-xs-block visible-sm-block"></div>

    <div class="col-xs-6 col-sm-6 col-md-3">
    Area 3
    </div>
    <div class="col-xs-6 col-sm-6 col-md-3">
    Area 4
    </div>
</div>
```

The result with the clearfix is shown below:

Area 1 Yada yada lorem ipsum dolor sit amet Londinium et Eboracum Novum et Muscovium etc etc etc
Area 2
Area 3
Area 4

Offsets

Sometimes it might be desirable to leave a gap in the layout. This can be done with ***offsets*** (see the documentation):

```
<div class="row">
    <div class="col-md-3 col-md-offset-3">
    After an offset of 3
    </div>
    <div class="col-md-3 col-md-offset-3">
    After another offset of 3
    </div>
</div>
```

In the example above, the first within the row is given a class of ***col-md-offset-3*** which means it will be offset 3 places from the left, i.e. it will occupy columns 4 to 6. The next ***again*** has an offset of 3, which means it will be offset 3 places from the previous in the row - i.e. it will occupy columns 10 to 12.

The results are shown below:

After an offset of 3
After another offset of 3

Navigation bars

Credits: this W3Schools article was used to research this topic in addition to the Bootstrap documentation.

Another useful feature of Bootstrap which allows mobile-friendly development is ***navigation bars*** which become invisible (accessible via a menu) on mobile devices. Here is an example of a navigation bar:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Bootstrap Test</title>
<link rel="stylesheet" type="text/css"
href="http://www.free-map.org.uk/course/wad/bootstrap-3.3.5-dist/css/bootstrap.min.css" />

</head>
<body>

<!-- The navigation bar -->
<nav class="navbar navbar-inverse  navbar-fixed-top">


<div class="container">

<!--stops the navbar moving to the left and becoming vertical on small screens-->
<div class="collapse navbar-collapse" id="navbardiv">

<!-- The actual set of links-->
<ul class="nav navbar-nav">
<li class="active"><a href="index.html">Index page</a></li>
<li><a href="http://www.free-map.org.uk/course/">DFTI</a></li>
<li><a href="http://www.free-map.org.uk/course/mad/">MAD</a></li>
</ul>

</div>

</div>

</nav>

</body>
</html>
```

This gives a navigation bar similar to what you see at the top of this page (but without the "Bootstrap Demo" title for the moment).

If we examine the HTML of the navigation bar above, it contains the following components:

- A <nav> element. This is the HTML5 semantic element for a navigation area (not specific to Bootstrap). We give the <nav> the following classes: ***navbar*** (which indicates it's a navigation bar); ***navbar-inverse*** (inverses the colour scheme, so white text on black background) and ***navbar-fixed-top*** (which fixes the navigation bar to the top of the visible screen even if the user scrolls downwards).
- A container . As we saw above, in Bootstrap, other elements should be placed in containers.
- A to hold the actual links. This has classes of ***collapse*** and ***navbar-collapse***. The purpose of these classes are to stop the navigation bar re-aligning to become vertical and to the left of the screen on small screens. If you don't care about that, you can leave this out.
- Finally we have the actual links, embedded within a list (<ul> and <li> elements). The <ul> must be given classes of ***nav*** and ***navbar-nav***.

Adding a hidden menu for mobile devices

The next step is to modify the navigation bar for mobile devices. A common setup for navigation is to have a "menu icon" on the navigation bar (consisting of three horizontal lines) which when tapped, makes the actual links appear as a submenu. Try resizing the browser window so that it's narrower for a demo. The following example shows how to do this:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Bootstrap Test</title>
<link rel="stylesheet" type="text/css"
href="http://www.free-map.org.uk/course/wad/bootstrap-3.3.5-dist/css/bootstrap.min.css" />

<style type="text/css">
.row div { border: 1px solid black;border-collapse: collapse; }
</style>

<script type="text/javascript" src="http://edward/ewt/jquery.min.js">
</script>
```

```
<script type="text/javascript" src="http://www.free-map.org.uk/course/wad/bootstrap-3.3.5-dist/js/bootstrap.min.js">
</script>

</head>
<body>

<nav class="navbar navbar-inverse  navbar-fixed-top">
<div class="container">

<div class="navbar-header">
<button type="button" class="navbar-toggle"
data-toggle="collapse" data-target="#navbardiv">
    <span class="icon-bar"></span>
     <span class="icon-bar"></span>
    <span class="icon-bar"></span>
</button>

<a class="navbar-brand">Bootstrap Demo</a>
</div>

<!--stops the navbar going to left on small resize-->

<div class="collapse navbar-collapse" id="navbardiv">

<ul class="nav navbar-nav">
<li class="active"><a href="index.html">Index page</a></li>
<li><a href="http://www.free-map.org.uk/course/">DFTI</a></li>
<li><a href="http://www.free-map.org.uk/course/mad/">MAD</a></li>
</ul>

</div>

</div>

</nav>

</body>
</html>
```

Note here:

- The hidden menu is implemented using jQuery and Bootstrap's own JavaScript. So we need to include both jQuery and Bootstrap's JavaScript library (bootstrap.min.js)
- Note the additional HTML code to implement the mobile hidden menu:

```
<div class="navbar-header">
<button type="button" class="navbar-toggle"
data-toggle="collapse" data-target="#navbardiv">
    <span class="icon-bar"></span>
     <span class="icon-bar"></span>
    <span class="icon-bar"></span>
</button>

<a class="navbar-brand">Bootstrap Demo</a>
</div>
```

  We create a div with a class of ***navbar-header*** to contain the menu. Then we create the button (the <button> tag) and give it a class of ***navbar-toggle*** and a ***data-target*** of ***navbardiv*** (this is the ID of the containing the links). This ***data-target*** attribute will link the menu button to the links, so that they appear when the user taps the button.
- Within the button we create three s with classes of ***icon-bar***. These represent the three vertical lines.
- We should also give the ***navbar-header*** a title so that the user can see the name of the application (for example, the "Bootstrap Demo" title on this page). This is done by creating a hyperlink with a ***class*** of ***navbar-brand***:

```
<a class="navbar-brand">Bootstrap Demo</a>
```

Exercises

1. Create a Bootstrap grid with the following rows:
    - First row should have two s in the proportion 1:5. They should contain the text "One" and "Two".
    - Second row should have four s in the proportion 1:2:2:1. They should contain the text "Three" to "Six" respectively.
    - Third row should on ***desktop*** devices have four s of equal width, but on ***mobile*** devices ***two*** lines each with two s in proportion 1:2. They should contain the text "Seven" to "Ten" respectively.
    - Fourth row should have a gap, a , another gap and another in the proportion 2:1:1:2. The two s should contain the text "Eleven" and "Twelve".
2. Create a Bootstrap layout for your mapping application. Create a labelled "OSM Map" and another containing your actual map. On desktops these should be in a single row of the grid in proportion 1:5. On mobiles the "OSM Map" label should appear ***above*** the map.
3. Add a mobile-friendly navbar to your mapping application, above the grid system. This should be labelled "OSM Map" and have the following links:
    - OpenStreetMap - www.openstreetmap.org
    - OpenCycleMap - www.opencyclemap.org
    - Freemap - www.free-map.org.uk

4. Try also adding the three links to your "OSM Map" in your grid. In mobile layout, the links should appear next to each other, and to the right of the "OSM Map" label. In desktop layout, they should appear vertically, and below the "OSM Map" label.

Further reading

This is just an introduction to Bootstrap, illustrating the most important topics for mobile-friendly development: the grid system and navigation bars. There are many other things you can do with Bootstrap; see the following:

- The Bootstrap site
- The W3Schools tutorials