BUILDING A MOBILE FIRST WEB SITE

AIM

In this project we are going to build a simple web site that consists of 5 pages. We are going to make the pages mobile friendly as well as viewable in older browsers.

The files you need to use are on Blackboard.

- Download the zip file of resources.
- Unzip the five pages *index.html*, *structure.html*, *presentation.html*, *interaction.html and contact-us.html*
- Open *index.html* in your editor and in Google Chrome

MOBILE FIRST STYLESHEET

A requirement of this site is that it is mobile friendly. In order to do this we will follow a technique known as 'mobile first'. This involves first designing a simple stylesheet for the site that will present the content in a mobile friendly fashion. The benefits of this are:

1. Focuses the designer on the key content.
2. The 'mobile' stylesheet will download first - other stylesheets for larger screens (laptops, desktops) will only be downloaded if required - as such this approach is kinder on mobile user's data allowance.

STYLING THE BODY AND CONTAINER

Create a file called *mobileFirst.css* in the *styles* folder. Add the following rules:

```
body{
	font-family: 'Open Sans', Arial, Helvetica, sans-serif;
	/* base font of 14 pixels - then use ems */
	font-size:14px;
	color:#545454;
	background-color: #ffffff;
	margin:0;
	padding:0;
}
#container{
	margin:0 auto;
	width:100%;
}
```

This sets the default font and colour for our pages as well removing the 'native' margin and padding that belonged to the `<body>`.

- Use the appropriate class and id selectors to pad the content of the `<section id="content">` and `<div class="boxContent">` 20px on the left and right hand sides.
- Also pad `<section id="content">` 10px from the top and add 20px to the bottom.
- Set padding to zero for the top and bottom of the `<div class="boxContent">`.

Tip: Use of the shorthand to set the box model properties.

STYLING THE NAVIGATION BAR

We will use a mix of HTML selectors, pseudo-classes and compound selectors to create full width buttons. This can be done with the following rules:

```
/* navigation bar  */
#menu{
	display:none;
}
nav {
	padding: 10px 0px 10px 0px;
	background-color: #15384C;
}
nav ul{
	margin:0;
	padding:0;
}
nav ul li{
	display:block;
	margin: 0 15px 15px 15px;
	padding: 5px;
	color: #a6abc5;
	border: 1px solid #ffffff;
}
nav ul li a{
	text-decoration:none;
	display:block;
}
nav ul li a:link, nav ul li a:visited{
	color:#ffffff;
}
```

STYLING THE HEADER

The design for the site requires a background image in the header bar. To do this we will create a rule that applies a background image to the header. The rule is as follows:

```
header{
      background: url(../images/smlBanner.jpg) no-repeat 90% 0px;
      background-size:600px 150px;
}
```

The background image is set using the `background` CSS shorthand. This has the following syntax:

background: [colour] [url] [repeat] [horizontal] [vertical]

In the Google Chrome inspector experiment with the values to see how the background image can be moved.

VIEWPORT

When this page is tested in the browser the background will appear to the right hand side. In Google Chrome use the Inspector to switch to the mobile view. Here the view is perhaps still not what was expected. This is due to the 'viewport'.

Web browsers on mobile devices are designed to cope with all web pages whether they have been made mobile friendly or not. To facilitate this, the mobile web browser behaves as if it has a much larger screen that it has in reality. Mobile browsers render pages in a virtual window known as the viewport - that is wider than the physical screen dimensions. However, if a page is designed to be mobile friendly, then the browser needs to be told not to use its default viewport but to use the one set by the designer. This is achieved through the addition of a meta tag to specifically control the viewport behaviour. Add the following to the `<head>` of the HTML file.

```
<meta name="viewport" content="width=device-width, maximum-scale=1.0,
minimum-scale=1.0, initial-scale=1">
```

This tells the mobile browser that the viewport is the same as the width of the actual device not the virtual viewport.

Now save and test your pages in the mobile emulator in Google Chrome.

STYLING THE LOGO

To style the `div#logo` create a rule as follows:

```
#logo{
	padding:10px;
	font-size:0.8em;
	background:none;
	color:#fff;
}
```

STYLING THE FOOTER

The footer at the bottom of the page includes a HTML entity to display the copyright symbol. Create a rule for the footer to centre the text and reduce its size slightly. It may appear as follows:

```
footer{
	padding: 0px 20px;
	text-align:center;
	font-size:0.8em;
}
```

Tip: always used HTML entities for the special character such as <, >, £, &.

SHOWING / HIDING THE MENU

Currently our navigation appears by default. However, we want to hide it and add a 'burger' menu to allow the user to access it. This will take up much less screen space.

To reveal the 'burger' button change the CSS for the #menu as follows:

```
#menu{
	display:block;
	margin:5px 0 10px 15px;
	padding:0 0 0 20px;
	background: url(../images/whiteNavicon.png) no-repeat 0;
	background-size: 15px 15px;
}
```

We should then hide the navigation bar. Add `display:none` to the `nav ul` rule.

We now need to add some Javascript (via jQuery) magic. Before the closing `</body>` add the following two lines of code.

```
<script src="js/jquery-3.1.0.min.js"></script>
<script src="js/main.js"></script>
```

FORMATTING FOR LARGER SCREENS

The CSS we've added now gives us a working design for mobile devices. However, if you view the page in Google Chrome in the 'normal' desktop the current CSS gives a poor experience to desktop users.

To fix this we'll add a second stylesheet that will only be used if the screen is greater than 601px.

As CSS rules are cascaded from one stylesheet to another the styling of the desktop view can be based on that of the 'mobile first' stylesheet. As such we'll be amended/overwriting some properties and in some cases adding new ones. The technique for doing this is called 'media queries'.

WORKING WITH MEDIA QUERIES

Media queries allow the designer to ask questions about the browser and device been used to view the page. If the query has a positive response then alternative stylesheets can be applied. For this project we'll ask whether the screen has a minimum width of 601px. If so, we will add another stylesheet to the page. Add the following to the `<head>` of the *index.html* immediately after the link to the mobileFirst.css stylesheet.

```
<link rel="stylesheet" type="text/css" media="only screen and (min-
width:601px)" href="styles/biggerScreen.css">
```

This is essentially the same as a normal `<link>` element but with the addition of the `media` attribute. The `media` attribute contains the 'query'.

Why 601 pixels? This is a design decision based on the current state of mobile devices. It will mean that some larger screen devices (especially in landscape mode) will have this stylesheet applied.

Test your page back in the 'normal' desktop view in Google Chrome. The page will appear with a dark blue background - this indicates the new *biggerScreen.css* is been applied.

TIDYING UP THE DESKTOP DESIGN

The *biggerScreen.css* has some rules but more are needed to create the desktop view.

First we'll hide the menu with:

```
#menu{
	display:none;
}
```

The `div#container` needs amending. Can you create a CSS rule that will give the container:

- margins of 10px at the top and bottom but auto for left/right
- a maximum width of 1200px
- a white background colour
- and rounded corners of 8px.

Add a new background image to the header and give it a height of 200px as follows:

```
header {
    border-radius:8px;
    border-bottom-left-radius:0px;
    border-bottom-right-radius:0px;
    height: 200px;
    background: url(../images/bigBanner.jpg) no-repeat right 0px;
    background-size: 1200px 200px;
}
```

ABSOLUTE AND RELATIVE POSITIONING

For the desktop design we would like to move the `div#logo` that holds the `<h1>`. This can be achieved a number of ways but we are going to use the `position` property. When the position property is set to a value of absolute then other properties such as top and left can be used to place the element in the page. With `position:absolute` the content is place 'relative' to its parent element. Try adding the following:

```
#logo {
    position: absolute;
    top: 100px;
    left: 500px;
    padding: 16px;
    text-align: center;
    border-radius: 8px;
    background: rgba(0,0,0,0.5);
}
```

Currently as there are no parent elements of `div#logo` that have a position property, the `div#logo` is placed absolutely in the browser window. However, if we add a `position:relative` property to the `div#container` this will make the `position:absolute` relative to the `div#container`.

Experiment switching these values on/off in the Google Chrome inspector to see the impact each property has.

Notice that the `<nav>` also have a property of `position:absolute`. By adding `position:relative` to the `div#container` the `nav` is also placed relative to the container.

ADDING COLUMNS WITH FLOAT

When in the desktop view we want the content to make use of columns. To do this we'll use the `float` property.

Before using float lets improve the padding on the content by using:

```
#content {
    padding: 60px 20px 10px 20px;
}
.three-column {
      padding: 0px 20px 15px 20px;
}
```

There are three `div.box` elements to float. Add the following:

```
.three-column .box{
      width: 33%;
      float: left;
}
```

The boxes will now 'stack' against the left margin each column taking up a third of the available space.

ADDING IMAGES TO THE TOP OF THE COLUMNS

One technique we could use to add images to the column head is the background property. If this is combined with appropriate padding values, we can add images to the top of each column. As this will be done purely through CSS, then the images won't appear in the mobile view (and won't be downloaded either). We want different pictures for each column therefore add:

```
#box1{
      background: url(../images/manhole150.jpg) no-repeat 10px 10px;
}
#box2{
      background: url(../images/art-deco-architecture150.jpg) no-repeat
10px 10px;
}
#box3{
      background: url(../images/vat-pic150.jpg) no-repeat 10px 10px;
}
```

… and then to ensure there is enough space for each image add the following with a large top padding value:

```
.boxContent{
      padding:170px 10px 10px 10px
}
```

MARGIN COLLAPSE

The footer in this design has the following rule:

```
footer{
      /* illustrates margin collapse */
    clear:left;
    padding: 40px;
    border-top: 1px solid #ccc;
    margin: 20px;
}
```

The `clear:left` rule is used to reset the float level such that the footer appears underneath the floated columns. The footer also illustrates an effect in CSS known as 'margin collapse'. Notice the `margin: 20px` value. Comment out the padding and border. Then edit the margin value. Try removing it. Try increasing it. It won't impact on the appearance of the page.

This due to a phenomenon in CSS known as 'margin collapse'. This is a complex browser behaviour where vertical horizontal margins are automatically 'collapsed'.

Vertical margins collapse so that:

- the bottom margin and top margins of siblings are combined when touching
- The top margins of parent and child are combined when touching
- the bottom margins of parent and child are combined when touching

When 'collapsed' the two margin values are combined to use the largest of the two. With parent/child combination the new margin value is then set on the outer element

To stop parent and child margins collapsing parents apply a border or padding value. This stops the two margins touching. Try toggling the padding and border values on the footer to see how the browser responses. Notice that the margin only works when either padding or border are set.

BUILDING A PRINT STYLESHEET

If a user decides to print our web page we can use media queries to apply a 'print' stylesheet. The `<link>` element to add a print stylesheet would appear as follows:

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

With a print stylesheet it is a good idea to reduce unnecessary images and colour, remove/reduce navigations bars and remove/reduce specific pixel based layout rules.

There are also specific printer properties such as `page-break-after: always` that can be used to force line breaks.

SUPPORT FOR OLDER BROWSERS

Older versions of IE do not understand the new HTML elements or media queries. There is a file called *html5shiv.js* in the *scripts* folder. This can be downloaded from:

http://code.google.com/p/html5shiv/

The SHIV/SHIM allows older browsers to at least style new HTML elements as blocks or inline as appropriate.

```
<!--[if lt IE 9]>
    <script src="scripts/html5shiv.js"></script>
<![endif]-->
```

To see what the page would look like in IE8 we can use the built-in emulator in IE11. In IE11 press F12 to open the developers panel, find the Emulation options and switch the document mode to '8'.

IE CONDITIONAL COMMENTS

The HTML5 shim makes use of IE conditional comments. These are HTML comments that older versions of IE interpret such that style rule can be applied to 'bug fix' issues in older version of IE.

This IE conditional commend `<!--[if lt IE 9]>` was used by the HTML5 shiv but we can also use it to add some styles that will only be applied if the browser version if it is less than IE9.

```
<!--[if lt IE 9]>
<script src="js/html5shiv.min.js"></script>
<link rel="stylesheet" type="text/css" href="styles/oldie.css" />
<![endif]-->
```

The *oldie.css* stylesheet is only loaded by Internet Explorer versions below IE9. We don't need to worry about the *biggerScreen.css* file as older versions of IE will ignore the media query and therefore won't load this stylesheet.

FIXING THE OTHER PAGES

The *index.html* page should now work on mobile, desktop and on older browsers. Add the necessary fixes to the pages, *structure.html*, *presentation.html*, *interaction.html* and *contact-us.html*.

This should include:

- The `<link>` tags for the stylesheets.
- The `<meta>` tag to set the viewport.
- The `<script>` tags for the menu button interaction.

BUILDING A FORM

Edit the HTML page called *contact-us.html* and add a HTML form with the following elements:

```
<form>, <fieldset>, <legend>, <label>

<input> - types text, email, tel, number, radio, checkbox, search, submit

<textarea>, <select>, <option>
```

When adding form elements it is important that they have a name attribute. This becomes essential later when the form values are submitted to be processed in some way - for example by a PHP script. The name attribute and the value entered by the user are referred to as a name / value pair.

For example the following creates a simple text field with a name of surname.

```
<input type="text" name="surname">
```

If the user enters a value of bloggs then the name / value pair submitted is surname=bloggs.

Tip: See http://www.mustbebuilt.co.uk/moving-to-and-using-html5/fabulous-forms/ for a review of the core HTML form elements you should get familiar with.

STYLING FORM ELEMENTS

Following the 'mobile first' approach create two additional stylesheets for the form.

The first stylesheet will style the form for mobile and then we'll produce a second stylesheet that will be applied to the desktop using media queries.

As such your styles for the form will be attached as follows:

```
<link rel="stylesheet" type="text/css" href="styles/mobileFirst.css" />
<link rel="stylesheet" type="text/css" media="only screen and (min-
width:601px)" href="styles/biggerScreen.css">
<link rel="stylesheet" type="text/css" href="styles/mobileFirstForm.css" />
<link rel="stylesheet" type="text/css" media="only screen and (min-
width:601px)" href="styles/biggerForm.css" />
<!--[if lt IE 9]>
<script src="js/html5shiv.min.js"></script>
<link rel="stylesheet" type="text/css" href="styles/oldie.css" />
<![endif]-->
```

With the limited space of the mobile is it common practise to place labels above form fields. As the `<label>` element is an inline element create a rule with `display:block` to force a new line.

Other design aspects to consider for the mobile stylesheet is the width of the form elements. Try to maximise this with `width:100%`.

For the desktop stylesheet try to align the labels to the left of the form elements. This could be done with `float`.