HTML / CSS – Lab Two - CSS Positioning

Objectives

1. Review the use of `<div>` elements in the layout of webpages
2. Understand the CSS float property
3. Consider the other CSS position options - `static`, `absolute`, `relative`
4. Understand the box model

Starter Files

Create a new folder for *week2* containing the starter files *index.html*, *structure.html*, *presentation.html* and *interaction.html*.

Open up *index.html* and review the HTML code. The file contains a series of `<div>` tags which are designed to give the pages structure.

Setting Up the Stylesheet

Create a CSS file called *main.css* and save it into the *styles* folder. Add a rule to set document defaults using the *<body>* tag as a selector.

```
body{
	font-family: Helvetica, Arial, sans-serif;
	font-size: 15px;
	color:#545454;
	background-color: #15384C;
	margin:0;
	padding:0;
}
```

As the content stretches across the whole width of the browser window we'll add a container to restrict it to a width of 1200 pixels. Set the margin to `auto` also ensures that margins are allocated equally to the left and right hand sides, which results in the centering of the container.

```
#container{
     width:1200px;
     margin:50px auto;
     background-color: #ffffff;
}
```

To target the header of the page we'll create a rule for the `<div id="header">` as follows:

```
#header{
     height: 150px;
     background-color: #cf4a41;
}
```

Aligning with float

Within the header we would like the navigation links on the right and the 'Web Application Development' text on the left.

In the HTML the `<div id="logo">` and `<div id="menu">` are siblings. As they are currently block elements `<div id="menu">` appears below `<div id="logo">`.

Add the following rule to illustrate how `float:left` effects the layout:

```
#logo, #menu{
     float:left;
}
```

Try experimenting with floating the `#logo` and `#menu` to the left and the right.

To position the `#logo` to the right of the `#menu` we can just use `float:left` but add some padding to the `#logo` to push the `#menu` further to the right. Remove the rule above and add two separate rules to replace them as follows:

```
#logo{
     width:590px;
     padding:60px 0 0 50px;
     float:left;
}
#nav{
     float:left;
     padding:60px 50px 0 0;
}
```

Converting Lists into Vertical Menus

Within the `#nav` there is an unordered list `<ul>` that contains all the `<a>` hyperlinks.

The bullet points of the lists can be removed with `list-style:none` attached to the `<ul>` and the list made vertical by floating each `<li>` to the left, causing them to stack next to each other.

```
#nav{
     float:left;
     padding:60px 50px 0 0;
}
#nav{
     list-style: none;
}
#nav li{
     float:left;
}
```

Targeting the `<a>` and their pseudo-classes we can remove the default underline but have it appear on user rollover.

```
#nav a{
     padding:7px 0 7px 30px;
     text-decoration: none;
     color:#ffffff;
}
#nav a:hover{
     text-decoration: underline;
}
```

Micro Managing Positions with position:relative

The contents of the `<h1>` needs repositioning slightly to improve the appearance of the header. Here we could use `position:relative` as this offsets an element from where it would appear in the normal flow. We'll also take the opportunity to set the font color to white:

```
#header h1{
     position:relative;
     bottom:15px;
     color:#fff;
}
```

Use the console in the browser to experiment with the bottom and/or top values to see how the text can be repositioned.

Relative vs Absolute

If the `position:relative` was changed to a `position:absolute` the `<h1>` would be plucked from the normal flow of the document. Any top, bottom, left or right properties would then, in this instance, be based on the entire document.

Any absolutely positioned element is placed relative to any parent node, that has itself a `position` value of either `relative` or `absolute`.

So for example setting the `#header` to be `position:relative` would allow the use of `position:absolute` to re-position the `<h1>`.

Note: All elements by default have the value of `position:static`. This means that they will remain in the normal flow of the document. A block element such as a `<div>` will stretch out horizontally as far as it can.

Creating a Sidebar with float

In the HTML file there is a logical division of `class sidebar`. This we would like to position to the right hand side of the main content that is contained in the `<div>` of id `content`.

In order to do this we will again use `float`. Add a rule for the `sidebar` as follows:

```
.sidebar{
	float:right;
}
```

The result is not really what we might have expected or hoped for. The sidebar does indeed appear on the right but it is underneath the main content. This is because the `<div>` of id `content`, which has no rule attached to it, behaves as a `<div>` will do by default which is to fill all the space horizontally as a block element.

One way to fix this is to use `float:left` on both the `content` and the `sidebar`. If the widths used are correct, this will have the effect of stacking the content against the left margin, which somewhat counter intuitively, will place the sidebar on the right hand side.

Amend the CSS as follows by adding a new rule for `content`.

```
#content{
	float:left;
	width:500px;
}
.sidebar{
	float:left;
}
```

There are still plenty of issues to deal with here as the footer now creeps up the document following normal flow and the white background is lost.

Using clear to 'reset' the float level

In order to move the footer back to where we expect it to be, this can be fixed by using `clear:left`.

The `clear` CSS property acts to 'reset' any existing floats in the file by not allowing floats in the given direction. These could be floats to the `left`, `right` or `both`. If we place `clear:left` on the footer element it will force the footer below any elements that have been floated to the left.

```
#footer{
     clear:left;
}
```

Note: the issue of the 'missing' background could also have been fixed by adding the property `overflow:auto` to a rule attached to `#page`. The background disappears as when all the children of an element are floated the parent does not have a height value. Another fix would be to add a `height` value to the `#page` - but a downside of that approach is the height value is not always known.

Understanding the Box Model

The footer now appears in a sensible position but because of the 500px width of the `<div id="content">`, the sidebar is appearing over the top of the image. The content `<div>` clearly needs to be wider, but the question is what value should this be?

The content and sidebar elements are within a parent element `<div id="page">` that has a width of 1200px. The `<div id="page">` has inherited this width from the `<div id="container">`.

As content and sidebar elements are within `<div id="page">`, their combined width cannot exceed 1200px or the sidebar element will be wrapped underneath the content `<div>`. To illustrate this add:

```
#content{
	float:left;
	width:900px;
}
.sidebar{
	float:left;
	width:301px;
}
```

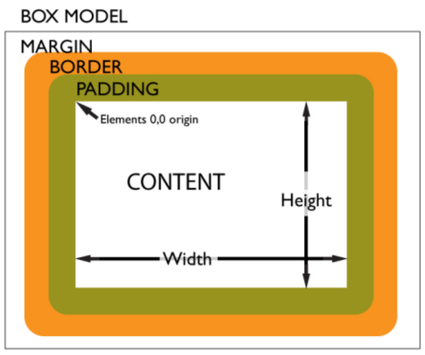
Even though the above is only 1px over the 1200px pixel width of the parent, the sidebar is wrapped.

To try and fix this, change the sidebar width to 300px.

This works for us but we could very quickly face another issue. The text in the content `<div>` is very close to the left margin. An obvious fix would be to some padding to the content `<div>` as follows:

```
#content{
	float:left;
	width:900px;
	padding-left:15px;
}
```

Unfortunately, the sidebar is again wrapped. This is where an understanding of the box model is important.



Note that the `width` property is that within the `margin` and `border` of any element. Therefore if we do the maths the 'true' width of the `div#content` is now 900px + 15px of left padding. Added to the 300px of `div.sidebar` gives a total of 1215px. This is 15px over the parent `<div id="page">` width so the sidebar is forced to wrap. Fix this by reducing the width of `div#content` to compensate for any padded added ie:

```
#content{
     float:left;
     width:885px;
     padding-left:15px;
}
```

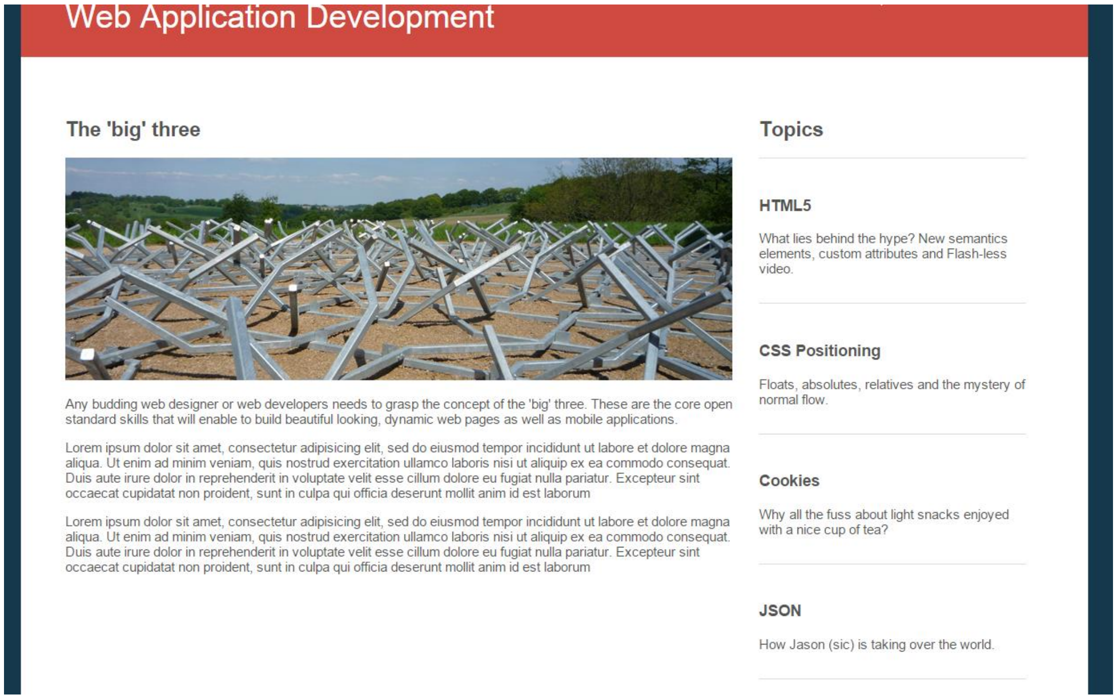
Experiment with the padding and margin values on the `content`, `sidebar` and `footer` to improve the appearance of the page.

Applying the Stylesheet to the other documents

Ensure the other files in the project are attached to the stylesheet by adding the following inside the `<head>` of each file.

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

Add to the stylesheet to present the sidebar in a more aesthetically pleasing way. Can you change it to appear as follows:



The 'big' three

Any budding web designer or web developers needs to grasp the concept of the 'big' three. These are the core open standard skills that will enable to build beautiful looking, dynamic web pages as well as mobile applications.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Topics

HTML5

What lies behind the hype? New semantics elements, custom attributes and Flash-less video.

CSS Positioning

Floats, absolutes, relatives and the mystery of normal flow.

Cookies

Why all the fuss about light snacks enjoyed with a nice cup of tea?

JSON

How Jason (sic) is taking over the world.

Tip: You'll need to use `border-bottom`, `padding`, `list-style` *and possibly the CSS pseudo-class* `li:first-child`.

Creating a Three Column Layout

In the file *presentation.html* there is a slightly different HTML structure.

```
<div class="three-column">
     <div class="box">
     …
     </div>
     <div class="box">
     …
     </div>
     <div class="box">
     …
     </div>
</div>
```

This is built for a three column layout. In order to achieve this, we can again use `float:left`. Create the following rule:

```
.box{
     float: left;
}
```

On its own this won't achieve much as we need to add a `width` property.

Bearing in mind the discussion above around the box model, and assuming that you would want three equal columns padded away from each other, extend the above to create a three column layout.