Week 4 Exercises

The aim of this week's exercises is to gain some experience in sizing and positioning elements; using borders, background images, opacity and text effects.

Time Required

These exercises should take you 1½ - 2 hours at the most, depending on your experience, and they are intended to be completed in class time.

Exercise 1: Floating elements

This week's exercises are first to experiment with floating two elements, and then to edit the stylesheet of another page for which I have already created the HTML.

1. Open a browser and locate the week 4 zip file on the headroomgate site at :
   `http://www.headroomgate.co.uk/style/downloads/week4.zip`

2. Download and unzip this file into your week 4 folder. You should see three HTML files and three CSS files along with a number of images.

3. Open the file **`ex1wk4.html`** in Dreamweaver and you will see there are two block elements inside the usual '**`container`**' div.
   a. If you view the file in Firefox you will see that the image is positioned above the (large) text of the paragraph, which is what you would expect to see (the grey border around the 'container' div is there just so that you can see the edge of that element).
   b. If you now go to the style sheet (in Dreamweaver)
      i. Give the '**`img`**' element a float instruction, i.e. **`float:left;`**
      ii. Look at the result
      iii. The text of the paragraph has surrounded the image to fill the available space, i.e. up to the grey border and has allowed the bottom of the div to come up.

It is not easy to see the fact that the *block* part of the paragraph element is actually underneath the floated image because the *inline* part of the element has only occupied the space around the floated element.

4. To see this give the paragraph a background colour of orange (**`#F90`**) in the style sheet, then look at the result. Two things you should note, the first is that the paragraph default style has inserted a top and bottom margin which shows as transparent space between the paragraph background and the '**`container`**' div. Second, the left edge of the paragraph, nothing seems to have happened around the image.

5. To show that the block part of the paragraph is underneath the image you should give the image a margin (say **`10px`**), this will create a space around the image which is transparent and without colour.

6. The colour of the block part of the paragraph element is there and showing as an orange block. If you now give the paragraph a margin of zero you should see all the transparent space around the text has gone.

7. If you go back and give the paragraph the instruction to float left can you guess what will happen? Try it and take a look.

8. Now try giving the paragraph a width of **`200px`** and view the result.

Can you explain what has happened here? When the paragraph was left without a declared width it assumed 100% of the width of the containing element and moved down to achieve that. However, when you gave it a width that could be accommodated in the available space next to the image so the paragraph moved there.

Did you notice that when the paragraph was floated without a width the border of the '**`container`**' div collapsed without any height and the paragraph appeared to be outside it? When you gave the paragraph a width it started at the same vertical point as the image, i.e. inside the top border of the '**`container`**' div. If you are not sure then take a close look at the position of the paragraph background colour, there is one pixel of grey above the colour and one pixel of grey behind the colour. You will see more about this in the next exercises.

Exercise 2: Creating the page layout

Open the file **`ex2wk4.html`** in Dreamweaver then take a look at the file in Firefox and you will see the complete but unformatted site that I have given the title 'Oxford City'.

1. Back in Dreamweaver open the stylesheet and you will find it is a standard new sheet.
    a. Start by giving the **`body`** element a font family of **`Arial`** etc
    b. Create an **`ID`** rule with the name of '**`container`**'
        i. Give this a width of **`700px`**
        ii. Top margin of **`zero`**
        iii. Bottom margin of **`40px`**
        iv. Use the margins to centre the '**`container`**' **`ID`** on the page
        v. Finally give the **`ID`** a background colour of white
    c. Save the file and take a look at the result in Firefox. The result should come as no surprise; the text is now in Arial and centred in the viewport.

2. There are three main structural divs on the page and you should begin by styling these so that you can start to get a feel for the page's layout.
    a. Insert an **`ID`** rule for the div '**`headline`**'
    b. Give it a 100% width
    c. Height of 170px.
    d. Insert a background image, the url for this is simply **`skyline.jpg`** as the image is in the same directory as the HTML, so the declaration is **`background:url(skyline.jpg);`**

3. Continue with the div which has the **`ID`** '**`mainBody`**' which only needs a width of 100%.

4. Finally the div with the **`ID`** '**`footline`**'
    a. Declare a width of **`96%`**
    b. Padding of **`1%`** and **`2%`**
    c. Background of **`#00C`**
    d. Text coloured **`white`** and sized at **`0.9em`**.

You should see quite a difference when you view the result in Firefox. There are still a couple of divs that need rules – '**`sidebar`**' and '**`content`**'.

5. Create rules for those **`ID`**s
    a. Give '**`sidebar`**' a width of 180px
    b. Give '**`content`**' a width of 478px.

c. Save the result and take a look in Firefox – no real difference yet, these last two divs need to **`float left`**, then what you see might surprise you – take a look and see if you can figure out what has happened?

That blue colour is coming from the footline which has applied its background rule from the last element that was not floated. To rectify this the '**`footline`**' div needs to cancel the earlier floats with the declaration **`clear:both;`**. If you type this then view the change in Firefox, the result should be more in line with what you might expect.

6. Before you leave these structural elements you should apply some padding to position the contents neatly within their divs. Give both '**`sidebar`**' and '**`content`**' padding of **`10px`** for both left and right and `zero` top and bottom.

One final bit of structural styling is to manage the placing of the two images in the main body of the page. These use divs with the class name of 'imageBlock'.

7. Create a class rule for 'imageBlock'
    a. Float it left with a width of 150px,
    b. Padding of 10px top and bottom, 42px left and right;
    c. Bottom margin of 10px.

If you look at the HTML you will see that the paragraphs forming the captions to the images are nested inside the image div, therefore a rule to style them needs a descendent selector.

8. Create a class with the descendent selector so: **`.imageBlock p { }`**
    a. Set the font size to 0.9em centrally aligned
    b. Set all margins at zero.

Take a look at the page again in Firefox and the images of the Radcliffe Camera and University College should side by side and roughly in the centre of the available (right hand) space with a small caption close to the image.

Exercise 3: Styling the body

At the moment the whole page has a white background which is rather clinical but more to the point you can't see where the container starts and finishes. You could just give the background to the page a simple colour but that's not very easy on the eye, you could give it a gradient colour so that those browsers that cater for gradient colours give the page a nicer appearance.

1. Locate the **`body`** rule in the stylesheet and add the following code:

```
background: #a1dbff;
background: -moz-linear-gradient(top, #f0f9ff 0%, #a1dbff 100%);
background: -webkit-linear-gradient(top, #f0f9ff 0%,#a1dbff 100%);
background: -o-linear-gradient(top, #f0f9ff 0%,#a1dbff 100%);
background: -ms-linear-gradient(top, #f0f9ff 0%,#a1dbff 100%);
background: linear-gradient(top, #f0f9ff 0%,#a1dbff 100%);
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#f0f9ff', ↵
endColorstr='#a1dbff',GradientType=0 );
```

To explain: the first declaration gives old browsers the instruction to make the background a mid-blue colour because they cannot render a gradient. The second declaration is the vendor specific coding for Mozilla based browsers, e.g. Firefox. The webkit version is for webkit based browsers, e.g. Safari and Chrome. The –o- targets Opera and the ms- targets Internet Explorer V10 and above. The penultimate declaration is the W3C standard and the final declaration is Microsoft's way of coding gradients for IE6-9.

If you save the changes and view the result you should see a gradual change of colour in the background as you scroll down the page. Now take a look at the page in Explorer, can you see a narrow border of the darker blue colour all around the edge of the viewport? That shows in all versions of Explorer up to IE9. This is because the default style for the body has a small margin applied and the margin takes on the default background setting, i.e. no gradient. You will want to turn it off I suspect. Return to the style sheet and give the **body** rule an extra declaration of **margin** set to zero. Save the file and look at the result, the border should have gone.

Notice in both browsers that even though you told the container div that it should have a zero top margin there is a gap between the top of the page and the viewport – why? You will take a look at that next.

Exercise 4: Styling the headline

The next part of the styling of this page is the headline. The current default style has the title hard over to the left in bold black with the Olympic link just an underlined piece of default text.

Have you worked out yet why that gap exists between the image and the top of the viewport? It is not very obvious but it is the styling of the h1 in the headline and it has a top margin set by default.

1. To see the effect open the stylesheet in Dreamweaver and insert a descendent selector of **#headline h1** { } and set the **margin** to zero.

Take a look at the result, you should now see that the gap has gone.

2. To continue styling the headline title (the **h1**) set the font to normal and 80px using the font shorthand (don't forget that this requires a font-family) give the colour as white and then view the result in Firefox. That looks a lot better but the text needs positioning better so you should apply a padding value – I gave it a **top padding** of **40px** and a **left** setting of **90px**; you might like to try different values to see what happens.

The title still lacks any 'life' and my way of dealing with that is to apply a touch of text shadow just to lift the text off the background – but this is a matter of taste. Try it out anyway even if you don't like text shadows!

The settings I used were:

```
text-shadow:2px 2px 2px #333;
```

The text-shadow property

The text shadow property allows for a list of values using the following syntax:

```
text-shadow: h-shadow v-shadow blur colour;
```

Where:

h-shadow	Required. The horizontal offset of the shadow. Positive values make the shadow on the right of the element, negative values move the shadow to the left
v-shadow	Required. The vertical offset of the shadow with positive values moving the shadow below the element and negative values moving the shadow above
blur	Optional – the distance of the blur
colour	Optional – the colour of the shadow which can be expressed in any of the CSS colour values, including rgba to set opacity

To develop your understanding of the possibilities I recommend that you try setting different values to see the results.

Now you should see what can be done with the Olympic link. I was imagining that the site would want to announce that the Olympic torch is going to pass through the city in July, and that there should be a link to the City Council's website, but currently you can't see it so it is not going to be very effective. The text is contained in a div called 'sticker' which has an anchor tag wrapped around it.

3. Open the stylesheet in Dreamweaver and after the '**`headline`**' **`ID`** create a new rule with the **`ID`** '**`sticker`**' and start by giving the div a **`width`** and **`height`** of **`80px`**.

When you view the results in Firefox you will see that the text has now spilt out into the main content block. Where I want it to go is over to the right of the header and upwards more towards the top.

4. To do this you should first **`float`** the **`ID`** to the right, then
    a. Give it a margin of **`-120px 20px 0 0`** notice that the top margin is a minus value; that is correct and it takes the div up from its start point by **`120px`**.

When you look at the result in Firefox you should see that the text has now gone to the top right area of the headline.

That is fine as far as it goes but it is still not looking very inviting, a bit of background colour would make the text have some substance; try **`background: rgba(255,255,0,0.8)`** and take a look at the result. What you should see is a yellow box containing the text but with a small degree (20%) of transparency. To improve this even further centre the text in the div (hint **`text-align`**) and take a look at the result.

You could leave it at that but you could also make the presentation better by giving the box some rounded corners. Try setting **`border-radius`** to **`50px`** and see the result. The circle is a bit too small for the text. If you recall from the lecture that adding padding enlarges the space taken up by the element, so try adding an overall padding of **`10px`** to this **`ID`** and check the result. The text is now comfortable spaced in a larger circle.

One final thing you could do to get the sticker noticed is give it a twist – you may remember this from week 1, the code is:

```
-moz-transform: rotate(20deg);
-webkit-transform: rotate(20deg);
-o-transform: rotate(20deg);
-ms-transform: rotate(20deg);
transform: rotate(20deg);
```

This is just one aspect of the 2D transforms available to you and it rotates the element through 20 degrees. **NB** the code above includes all the vendor-specific prefixes before the standard declaration.

Before you leave the headline you should attend to the anchor tag as some browsers will still style the text as a child of the div, which looks unpleasant.

Create a rule for all anchor tags and set it to a colour of **`#009`** with no underline.

Exercise 5: The sidebar

Turning now to the sidebar: first there is the issue of the images; currently they are off to the left except for the padding in the div, and the captions are larger than is appropriate for sidebar details.

1. The images are contained in divs with a class of '**`sideImg`**'
    a. Create a new rule for that class
    b. Set the width to 150px
    c. Give it a padding of zero
    d. Set the margin such that the div gets centred horizontally with
        i. Top margin of 20px
        ii. Bottom margin of 10px.

Take a look at the result. That sets up the positioning nicely. Now to set the text size for the caption; this is in a paragraph inside the '**`sideImg`**' div. You will require another descendent selector; can you work out the syntax? When you have created the new rule for the paragraph within the '**`sideImg`**' div set the **`margin`** and **`padding`** to **`zero`** and the **`font size`** to **`0.7em`**.

View the result in Firefox and you should find that the captions are close to the images and suitably small enough not to be too distracting.

Finally, before you go to work on styling the main content area you need to do something with the Tourist Information telephone number. This contained within a div with the ID '**`tourist`**'.

2. Create a rule the ID '**`tourist`**' and start by setting
    a. Width to 140px
    b. Padding of 12px
    c. Text colour to white
    d. Background to #004C92.

Take a look at the result in Firefox and you should see a rather chunky blue box with white text which is effective but not attractive in my view.

3. Centralise the text and then
    a. Centralise the div
    b. Top margin of 20px
    c. Bottom margin of zero.

        That looks a lot better but is still just a block.

    d. Try rounding the corners, you already know the syntax for this just try increasing the value until the corners join up and you have a sausage shape. There is no right or wrong answer to this, just a degree of corner rounding that suits you.

Exercise 6: Finalising the content

You have now styled the whole page except for some elements of the main page content. I am not very keen on the size and weight of the h2 here so I styled it as a weight of normal and gave it a size of 1.2em. The last change I made was to the paragraph in the section on the economy. This paragraph has a class of 'display'.

1. Create a new rule for class '**`display`**' and start by giving it a
    a. Width of 80% and a

b. Background colour of #e6e6fa.

   When you view the result in Firefox it is not very attractive at all at the moment as the text is tight against the edge of the colour of the block.

c. Setting the padding to 20px should make that look less claustrophobic when you view the result.
d. Now place the element in the centre of the available space and give it a top margin of zero and a bottom margin of 20px.

That looks much better but you still have one more style to introduce, the box shadow.

e. To achieve this you will need to add another declaration with the property name of **`box-shadow`** with values set to **`5px 5px 8px 1px rgba(0,0,0,.3)`**.
f. Place these details in your stylesheet and look at the result, then come back for the explanation.

The box-shadow property

The box shadow property is very similar to the text-shadow property you met in exercise 3 although the box-shadow property allows you to do more with the shadow. The property for a comma separated list of values using the following syntax:

```
box-shadow: h-shadow v-shadow blur spread colour inset;
```

Where:

h-shadow	Required. The horizontal offset of the shadow. Positive values make the shadow on the right of the element, negative values move the shadow to the left
v-shadow	Required. The vertical offset of the shadow with positive values moving the shadow below the element and negative values moving the shadow above
blur	Optional – the distance of the blur
spread	Optional – the size of the shadow
colour	Optional – the colour of the shadow which can be expressed in any of the CSS colour values, including rgba to set opacity
inset	Optional – changes the shadow from being outside the element (i.e. outset) to a shadow inside the element

To develop your understanding of the possibilities I recommend that you try setting different values to see the results.

Exercise 7: Collapsing margins

To demonstrate the consequences of collapsing margins open the file **`ex7wk4.html`** in Dreamweaver and you will see a level one heading and a paragraph, both containing text. Below that you will see 5 nested divs with some text in the centre. These should be left commented out for the moment.

If you look at the styling of the h1 you will see that it has zero margin, thus it is not having any vertical effect but the colour of the background extends all the way across the page. The paragraph has a top margin of 20px and you can clearly see that the paragraph is separated from the h1 by 20 pixels, which is what you would expect.

If you return to the stylesheet and give the h1 a bottom margin of 25 pixels then view the result you might expect to see margin between the two element increase by a further 25 pixels. Save the updated stylesheet and view the result in Firefox.

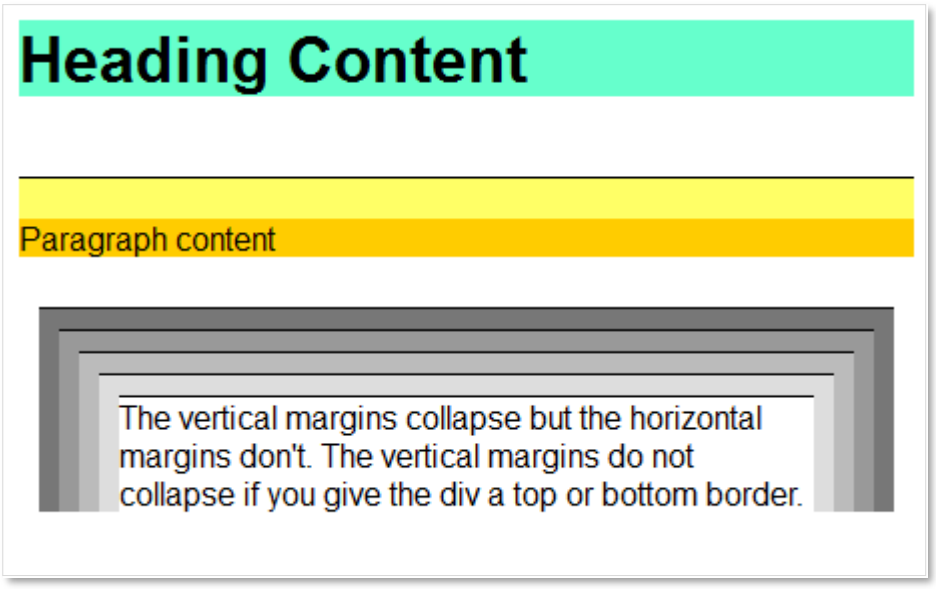
You will have seen only the very smallest change; this is because the effect is not cumulative – unless one or more margins is a negative value. To test this out change the top margin of the paragraph to -20px and view the result. You should find the margin has collapsed to the 5px difference between the two margins, i.e. 25px + -20px = 5px.

The behaviour between parent and child elements is the same, i.e. margins that 'touch' collapse, i.e. where there are no padding, borders or content between them. The best way to understand this is probably through an example. Return to the HTML and wrap a div around the paragraph:

```
<h1>Heading Content</h1>
<div>
<p>Paragraph content</p>
</div>
```

That makes the paragraph the child and the div the parent. Return to the stylesheet and give the div a top margin of 40px and a bottom margin of 25px. Reset the paragraph's top margin to 20px. If you now view the result in Firefox you will see that the space between the heading and the paragraph is only 40px, the result of the div and paragraph margins collapsing to the greater value – the 40px top margin of the div. Now introduce a border between the parent and child by giving the div a border-top of (say) 1px, solid black then view the result. You should see that because the border comes between the parent and child's vertical margins so they both get honoured. To explain the display: the green background denotes the content of the h1 which also has a bottom margin of 25px. The orange shows the content of the h1 and the yellow shows its top margin by displaying the content colour of the div. the white space above the border of the div shows the top margin of the div collapsed with the bottom margin of the h1.

To see a more comprehensive visualisation of collapsing margins return to the HTML and remove the comment from around the nested divs. When you view the result it should look like this:



If you turn off the top border of the class 'box' in the stylesheet you should see that the margins collapse and you only see the effect of the horizontal margins. If you add a border to both the top and bottom of the 'box' class you should see that the margins don't collapse because there is something in the way, i.e. the border.

To see more of this example visit **`http://reference.sitepoint.com/css/collapsingmargins`**