School of Computing, Maths & Digital Technology
Division of Digital Media & Entertainment Technology

Unit 6G4Z2101: Introduction to Web Design and Development

Worksheet 3: Styling links for navigation and Using CSS Position

Workshop topics

You will learn how to
- Style links for navigation
- Apply “display: inline” to change from block display
- Apply CSS position

Now that you have learned how to create lists in an HTML document, and how to apply CSS styling, you can create different styles of navigation menu.

Getting Started - Download and save files

Extract the contents of the navigation.zip folder in this week’s Moodle area into your own network folder. The folder contains a simple web page and a style sheet.

Style links as a navigation menu

1. Open the two files in Notepad++ and add the required code to the html document to link it to the style sheet (refer to the lecture slides in last week’s Moodle area).

2. Examine both documents carefully. Look at the tags in the html document, and examine the styling information for each part.

3. Open **navigation.htm** in a browser. On the right you have a list of links which are all linked to external websites about web usability and accessibility. The **list items** have a style to set the text size, but otherwise they’re displayed using the browser defaults. Because it’s an unordered list of hyperlinks, the default display is bullet points, and the link text is underlined as the default display for anchor tags.

   Now you’ll style the navigation menu. You only need to work on the css document; the code in the html document remains the same as it is already tagged.

4. Remove the default underline from the hyperlinks, using the **text-decoration** property with a value of **none**. Consider which selector you will need to target.

Save the css and refresh the html in your browser to check that the underline has gone.

5. Add a style to remove the bullet icon from the list items. You should know how to do this from previous labs.

   When you style a list of links, you need to think about what selector to choose as you apply your styles; the anchor tag **<a>**, the list **<ul>** or the list item **<li>**. With CSS there is often more than one way to achieve the same end result. You will compare the different effects of applying a border to an inline element (the **anchor** tag) and to a block-level element (the **list item** in which the anchor tag is contained)3.

6. In your CSS document add a **bottom border** to the **anchor tags**.

7. Save the css and refresh the browser. You'll have much the same effect as when the links had their default style; the border looks just like underlining, and only displays under the text characters. This is because the anchor tag is an **inline** element and only takes up the width of its **content**, whereas a block-level element fills the width of its **container**.

8. Add the following property and value to the anchor tags:

   display: block;

9. View in a browser and the bottom border will now reach to the right-hand edge of its container, because you have just changed it to behave like a block-level element.

10. Remove **display: block;** then move the border style from the **anchor tag** selector to the **list item** selector.

11. Check in a browser and the border will look the same. This is because the list item is a block-level element by default. So this is the better tag to select for the border. But why does the bottom border not extend to the left-hand edge?

    By default a list is given some default padding by the browser. Add a style to set the left-hand padding of the **list** to zero. This will allow the bottom border to extend across the full width of the nav container.

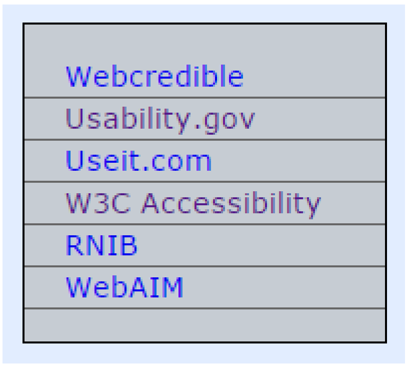
12. Add appropriate styling to move the navigation link text away from the left-hand border of its container, but make sure the border remains full-width.

NOTE:

You can find out more about the difference between block and inline elements at http://www.impressivewebs.com/difference-block-inline-css/.

13. Your menu should look like the image on the right.

    There is a double border at the bottom, because the list items have a bottom border and the nav container has borders around all 4 sides.



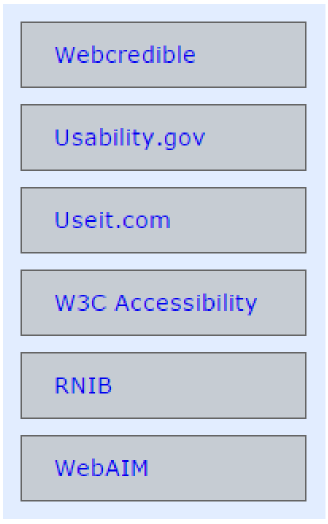
14. Remove the **border** statement from the nav container and apply it to the **list items** instead.

    You will find that the navigation background colour extends outside the border. This is because it's applied to the **nav** container, which occupies more space than the unordered **list**.

15. Move nav's **background colour** statement, and apply it to the **unordered list** instead. This should solve the background colour overspill.

16. Make adjustments to your stylesheet to make your menu look like the screenshot on the right. You will need to consider margins, padding and the background colour.

    If necessary refer to last week's lecture notes to see how margins and padding work in the CSS Box Model.

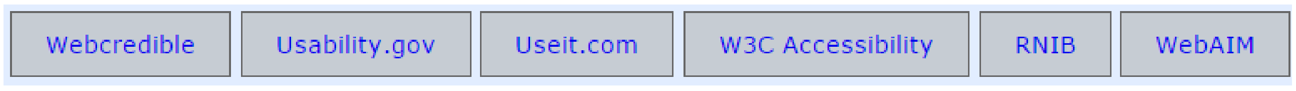


Create a horizontal list

1. Make sure your CSS file is saved. Use **File > Save As** and save it as **style2.css**.

2. Change your html document so it links to this new file.

3. You can easily change your CSS document to make your menu display like the screenshot below. It can be achieved in two different ways, as outlined on the next page.



4. First, change the width of the navigation container to **auto** so it can stretch to allow the links to spread across the page. Try the following two methods for turning your vertical navigation menu into a horizontal one.

5. Add the statement **display: inline;** to the **li** selector. This tells the browser that each list item should only take up the width of its content, plus any margins or padding.

Exercise 2 – Styling to improve usability

It is good practice to show users which link they're hovering over, which links they have visited, and which link they are pressing by adding a standard set of styles.

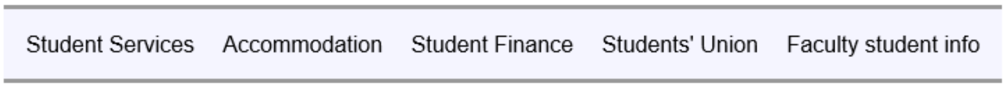
1. Refer to http://www.w3schools.com/cssref/sel_hover.asp, and add 'link', 'visited' 'hover' and 'active' styles to your stylesheet.

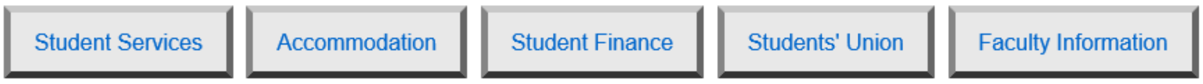
2. To make it even clearer which link the user is hovering over, make the 'hover' style apply to the background rather than the text of the anchor tags.

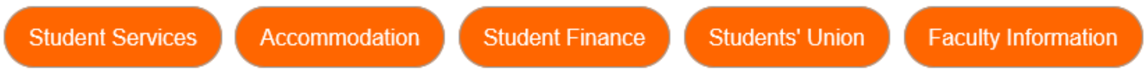
Try it on your own

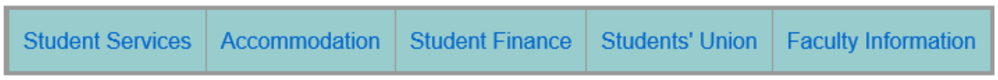
There are some example styles below. Experiment with creating at least two other navigation menu styles, but try more if you can. You can make a copy of your html and css files to so this, keeping the ones you have just completed. You are just applying different settings with the css box model in mind; thinking about borders, margins and padding.

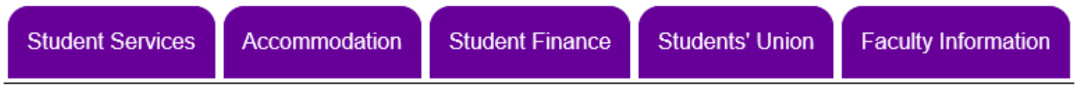
For rounded corners you use **border-radius: 15px** (changing the pixel value to give the size of curve required).

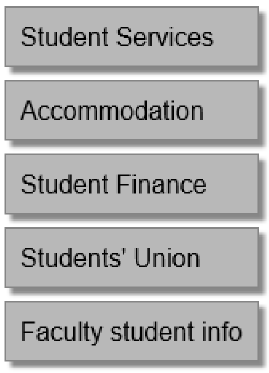












Applying CSS Postion

Go to this week's Moodle area and get a copy of the position.zip file. Save it to your own network area and unzip the contents. The file contains three web pages and an image.

Create the html document structure

1. In Notepad++ open the file called **css_overview.htm**. You will see that there a couple of headings and a few paragraphs of text.

2. Add the standard html tags in appropriate places, i.e. <doctype>, <meta>, <html>, <head>, <title>, <body>, and their closing tags.

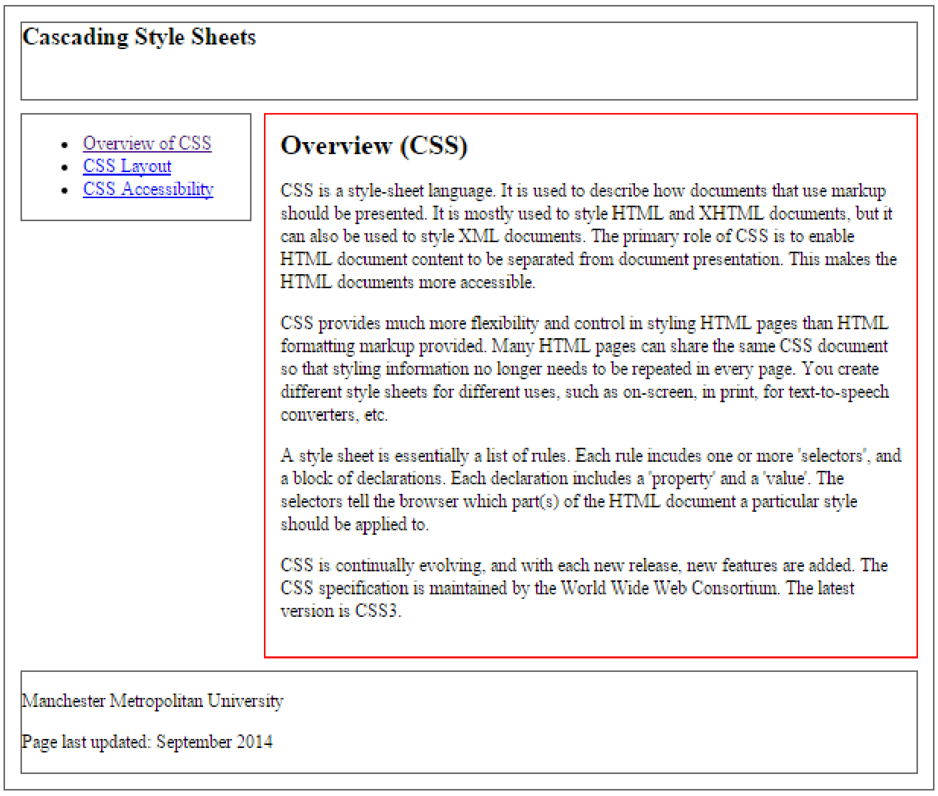
3. Using the diagram below for guidance, do the following:

    a. add <header>, <article> and <footer> tags to the html

    b. add a div called "container" and its closing tag. The tags should surround all the page content.

    c. add a div called "InnerContainer" and its closing tag. The tags should surround the **nav** and **article** boxes.

    d. add **nav** tags and inside them create a list which contains internal links to the all three html pages (there are two more in the folder).



Cascading Style Sheets

- Overview of CSS
- CSS Layout
- CSS Accessibility

Overview (CSS)

CSS is a style-sheet language. It is used to describe how documents that use markup should be presented. It is mostly used to style HTML and XHTML documents, but it can also be used to style XML documents. The primary role of CSS is to enable HTML document content to be separated from document presentation. This makes the HTML documents more accessible.

CSS provides much more flexibility and control in styling HTML pages than HTML formatting markup provided. Many HTML pages can share the same CSS document so that styling information no longer needs to be repeated in every page. You create different style sheets for different uses, such as on-screen, in print, for text-to-speech converters, etc.

A style sheet is essentially a list of rules. Each rule incudes one or more 'selectors', and a block of declarations. Each declaration includes a 'property' and a 'value'. The selectors tell the browser which part(s) of the HTML document a particular style should be applied to.

CSS is continually evolving, and with each new release, new features are added. The CSS specification is maintained by the World Wide Web Consortium. The latest version is CSS3.

Manchester Metropolitan University

Page last updated: September 2014

Add a stylesheet and basic styling

1. Create an external stylesheet and save it to the same folder as the html documents.

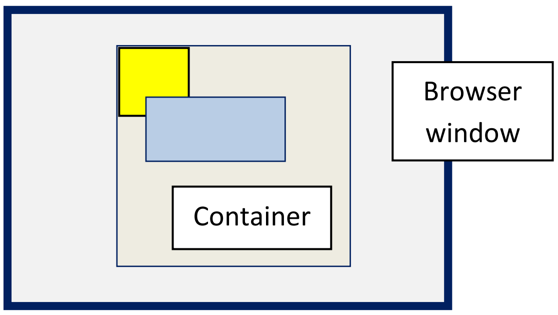
2. Add code to your html document to link it to your stylesheet.

3. In your stylesheet, add a style which clears (sets to zero) all the browser's default **margins** and **padding** for the following: body, h1, h2, div, nav, header, footer, article, ul, li. You can do this in a single statement.

4. Set the **font-size** for **body** to **14px**.

In worksheet 2 you used the float property to position elements next to one another. Now you'll use relative and absolute positions. You will be given some code to get started, but then you should work out statements and values out for yourself.

Relative position positions an element relative to the top left corner of its containing element.



The position of the container is set to "relative", so the yellow and blue boxes are positioned relative to the container's top left corner, rather than being relative to the screen's top left corner.

The yellow box is at the position of **top: 0;** **left: 0;**

The blue box is at the position of
top: 60px;
left: 20px;

1. Start by applying a 'css reset', which selects all the elements on your html page and sets their **margin** and **padding** to **0** (zero). Refer to last week's lecture slides.

2. Set the following border for #container, header, nav, article, footer :
   border: 1px solid #666;

3. Add the following code to your stylesheet, giving the container a width, height, top margin and centring it horizontally:

```
#container {
        position: relative;
        width: 800px;
        height: 600px;
        margin: 30px auto;
        border: 1px solid #666;
}
```

4. By setting the position to relative, any containers nested inside it will be positioned relative to its top left corner.

5. In your stylesheeet add a new style for header, and set the position as shown below. This will give the header a specific pixel location and size, leaving a 10px margin on each side.

```
header {
        position: absolute;
        top: 10px;
        left: 10px;
        right: 10px;
        height: 50px;
}
```

Container width = 800

Width of header is 780, added to 10px on the left, leaves 10px on the right

Next you will position the navigation container. First you will give it a red border so that you can see it clearly.

6. Give **nav** a **1px solid red border**. In the browser you will see that nav currently fills the width of the container (as it's a block-level element).

7. Give **nav** an **absolute** position so that it sits **10px** below the header, and **10px** in from the left and right.

8. Change the selector for the red border from **nav** to **article** so you can see the article's boundaries clearly.

9. Give **article** an absolute position so that it sits **10px** to the right of **nav** and level with the top of **nav**. The right-hand border of article should sit **10px** inside the right-hand border of the outer container.

10. Position the **footer** so that it sits **10px** away from the **bottom** of the container, and **10px** in from each side.

As there is no height set for the article box, which is the main content of the page, this area expands in height to fit its content. . To maintain consistency across different pages, you could fix the height of this area or remove the bottom borders.

There are a couple of issues with this sort of very fixed layout:

- If the stylesheet is attached to other pages, you would always need to ensure that the content of article is never longer than will fit inside the box
- If you reduce the size of the browser window there is no resizing of the page, so content is hidden.

If your content and layout are simple, as they are here, you can change the width of the container to a percentage, and it will enable text to wrap with smaller browser windows. However, it will get to a point where there is overlap at the bottom, so you could also apply a min-width.

Where layouts are more complex, box sizing is often fixed, but layouts can be made responsive to media queries to improve the user experience.

Further development

1. Change the container width to 85% and note that the main content will now word-wrap.

2. Change the layout so that the menu spans horizontally across the page, with main content underneath.

3. Add styling for paragraphs and headings (size, colour, font-family, perhaps line-height). For size use **em**. 1em is equivalent to the 14px font size you set.

4. Experiment with using percentage values for widths and heights, to make the page more scalable. Remember from the lecture that if you apply percentage height with relative position, any containing elements must also have a height (because percentages are calculated based on the parent element's dimensions). Remember that if your page scales down well to a certain point, you can apply a min-width to stop it being resized beyond a certain point.

5. Try splitting the main content area so that there are two equally sized boxes, horizontally aligned.

6. There is an image in the folder you downloaded. Add it to the text area and make the text wrap around it. Add appropriate styling to create some space between the image and the text.

7. You currently have a very basic design and layout, with borders around all the content boxes. Add your own styling to improve the design.

8. In your folder there are two further html pages. Link them to your css document, then tag them up appropriately to apply the styles from the style sheet. Add navigation links to connect all the pages.

   You can find out about other ways to control potential content overflow on lynda.com (link below).http://www.lynda.com/CSS-tutorials/Controlling-content-overflow/86003/97828-4.html