# CSCU9B2 Practical 3: Semantic HTML & CSS

## Aims:

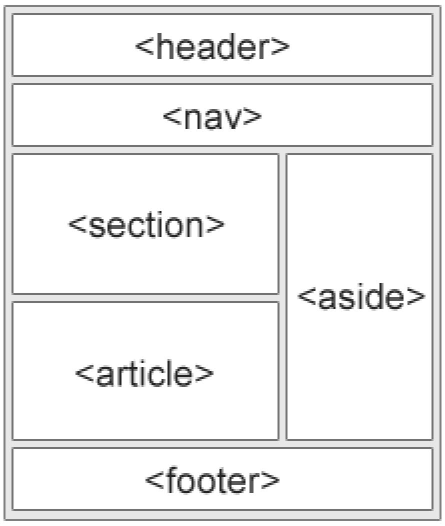
- **To use the new semantic HTML5 tags for structuring a page.**
- **To use CSS properties to lay out a page.**

**Please register your practical attendance: Go to the GROUPS\CSCU9B2 folder in your Computer folder and double-click on the Register icon. Ask a demonstrator if you need help or something goes wrong.**

**This sheet contains one checkpoint (see end of sheet).**

## HTML5 Semantic Tags and Page Layout

Semantic elements are elements that describe their content and meaning to both the web browser and developer. Some examples of semantic tags are the <img> and <table> tags that we have seen in the previous practicals. HTML5 introduces a number of new semantic tags to define different parts of a web page, such as the page header, footer, set of navigation links, etc. These include the <header>, <nav>, <footer>, <section>, <article> and the <hgroup> tags.



Before the HTML5 structural semantic tags, developers used code like:

```
<div id="my_nav1"> … </div>

<div class="my_header"> … </div>

<div id="this_footer"> … </div>
```

to indicate navigation, header, and footer of a web page. The id attribute names were quite arbitrary, which made it impossible for search engines to identify the correct web page content. With elements like: <header>, <footer>, <nav>, <section>, <article>, this becomes much easier.

## Adding sematic tags

To try these tags out, first, copy folder **BackToLatin** from **Wide\Groups\CSCU9B2** to your Web folder. This folder contains an HTML file and two images. You will use the new HTML5 semantic tags to mark up the structure of the file **back_to_latin.html**. Open this file in TextPad (or your usual editor; choosing Configure->Word Wrap will be useful for viewing it in TextPad) and do the following.

Place the first three headings, as well as the first list of navigational links, in a <header> element. The headings need to be grouped together with the <hgroup> element, while the navigational links should be placed inside the <nav> element.

The main text and headings need to be contained in two <article> elements, which are placed inside a <section> element.

Finally, the copyright information must be placed inside the <footer> element.

## Adjusting the page layout using CSS

Now that you have structured the page with the new HTML5 semantic tags, the next step is to apply some styles and layout to it. Add an embedded style sheet to the file to do the following.

Apply a different background colour to the three parts of the web document: <header>, <section> and <footer>. This is achieved with the property **background-color**, e.g.;

```
header
{
   background-color: yellow;
}
```

Now position these three elements in the centre of the browser window. The syntax is e.g.,

```
header
{
    width: 80%;
    margin-left: auto;
    margin-right:auto;
}
```

Push the images to the right using the **float** property, allowing the text to wrap around them. Furthermore, include a border around the images with the **border** property.

**Advanced:** Present the links of the <nav> element in one line using again the **float** property, and position them at the centre (this can be done with the **margin** and **width** properties, as above). Include some space between the links with the **padding-left** property. Get rid of the bullet points by setting the **list-style-type** property to none. You will probably have to add the "**overflow: auto;**" property to the containing <header> otherwise the list items may float out of the surrounding header box!

## More Page Layout with CSS

In the dim and distant past, page layout was done using tables. While this does work, it was never intended that tables be used in this way. In HTML5 (using CSS3) there are better alternatives.

In CSS, the **position** property of a tag can be set to one of four values:

- **static**: the default value, and the normal order of things, as they appear in the HTML itself
- **absolute**: the element can be placed in any position, using top, right, bottom, and left
- **relative**: the element can be placed offset from where it would have been, and the location is set using top, right, bottom, and left (again)
- **fixed**: is like absolute, but this time the location is relative to the browser window (and not the web page).

For example, assume we have an image with id *myimage*, perhaps generated by the HTML:

```
<img src=”abcdef.jpg” id=”myimage”>
```

In the absence of any CSS this image will arrive in a location dependent on the HTML before it, such as simply in line with any surrounding text. But let’s say we wanted to place it at the bottom left of the web page: the following CSS will achieve this:

```
#myimage
{
position: absolute;
bottom: 0px;
left: 0px
}
```

Note that replacing **absolute** with **fixed** fixes the image to the bottom of the browser window, even if not all the web page is displayable in the window!

A similar effect can be achieved with text, most easily using <div> tags, or directly in the <p> tag.

First, generate a web page with three paragraphs (each of a reasonable length, say 100 words or more), and two images (you could use some of the content of back_to_latin.html). Place the images at the **top left** and **top right**. Notice that if you don’t specifically place the text somewhere, it will end up being behind the images. So you need to adjust where the top of the text should be.

If, in addition, we use the **width** attribute, we can use this technique to allow generation of multicolumn text. This is often used in web pages: a narrow column of text on the left (say 150 px wide), and another column whose width is the rest of the browser window. Adjust your HTML so that what is displayed is the two images at the top left and right, with one paragraph displayed in a narrow column on the left, and the other two paragraphs on the right. Ensure that the text is not obscured by the images.

Note that because we can now have more than one element of the HTML displayed in one place, we might need to consider which one should go in front. This can be achieved using the z-index property: this has the value 0 by default, so setting it to 1 generally ensures that this object is displayed in front.

So: add a short piece of text that is displayed, in large red font in front of the left image.

## CHECKPOINT [HTMLCSS]

Please show the lab demonstrator the **back_to_latin.html** page with the structured content using the new HTML semantic tags. This page also needs to have the basic specified page styles and layout. Also show the demonstrator the page with images at the top left and right, and the text in columns, as described above.