

Making Basic Web Pages

The program is made of 10 modules M1 to M10 with each module and associated files in a separate folder.

M1 -What is HTML?

HTML is a **markup** language for **describing** web documents (web pages).

- HTML stands for **Hyper Text Markup Language**
- A markup language is a set of **markup tags**
- HTML documents are described by **HTML tags**
- Each HTML tag **describes** different document content

Overview of HTML

- Hypertext is a way of organizing and presenting information
 - Allows you to link documents in a non-linear way
 - Document contains links, or pointers to other information such as other pages, images, sounds, etc.
- Hypertext is a markup language
 - Tags determine structure and formatting of document
 - Subset of SGML (Standard Generalized Markup Language)
 - HTML (Hypertext Markup Language)

HTML tags are **keywords** (tag names) surrounded by **angle brackets**.

`<tagname>content</tagname>`

- HTML tags normally come **in pairs** like `<p>` and `</p>`
- The first tag in a pair is the **start tag**, the second tag is the **end tag**
- The end tag is written like the start tag, but with a **slash** before the tag name

An HTML Web Page.

Every file type has a file name made up of two parts – a “Name” and an “Extension” separated by a period. – Example: mywebpage.html

The file extension tells the computer what software (Application) is usually needed to open the file.

Typical examples are:-

- MyLetter.**doc** - Use Microsoft Word
- Annualtaxes.**XLS** - Use Microsoft Excel
- Report.**PDF** - Use Adobe Reader
- Webpage.**html** - Use a browser like Internet Explorer or Mozilla Firefox

BUT

Other applications (programs) can be made to look inside any file.

Creating an HTML document

- Each HTML document you create is at least one file. When it is displayed by a browser, it results in one page (may be scrollable so it can be as long as you want it to be.)
- It must be saved in text format.
- If you use “Notepad” as your text editor, you will get text format automatically.... There are many other HTML editors out there that provide enhanced built-in functions. We will start by using Notepad where you have to type everything in by hand.
- By convention, all HTML filenames are written in lowercase
- All HTML documents must end in the extension “.htm” or “.html”

–Example: myhomepage.html

File location and layout for the class.

The files used in this class are located in the “Library\Documents\My Documents” area of the computer that you are using in a folder initially labeled “Web_Class”. Please change this file name to *Your Name* (First and Last joined by an underscore).

Inside this root file, you will find some sub-folders:-

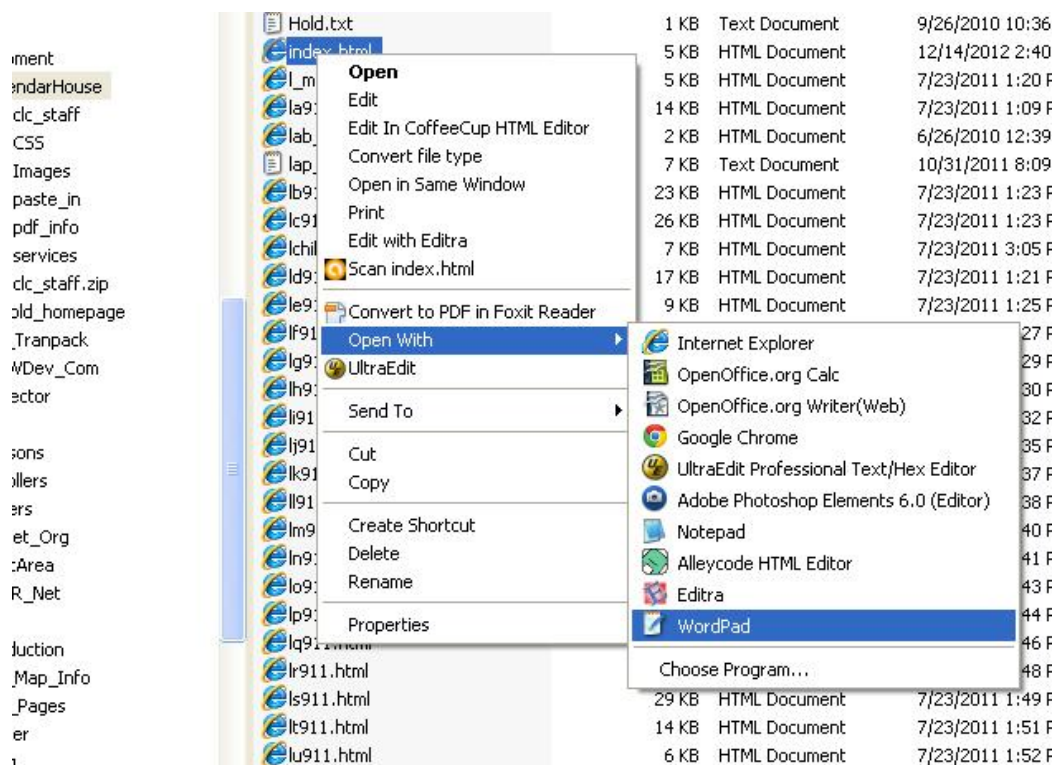
- **ClassWork** which in turn contain further sub-folders called M1, M2 ,M3 etc each of which contain the files for each **Module** of the class. They are separated this way solely to keep the various files in some semblance of order.
- **Links** which will take you to various reference places
- **Software** which we will be using later in the class
- **WWW** which we will rename uniquely for each student and which will contain the web pages that you will produce for upload to the internet.

Working with Web Files

Normally, you would open a file by double clicking on it. This would typically cause the operating system to look at the file extension to find and start the default application needed to show the file in its intended manner.

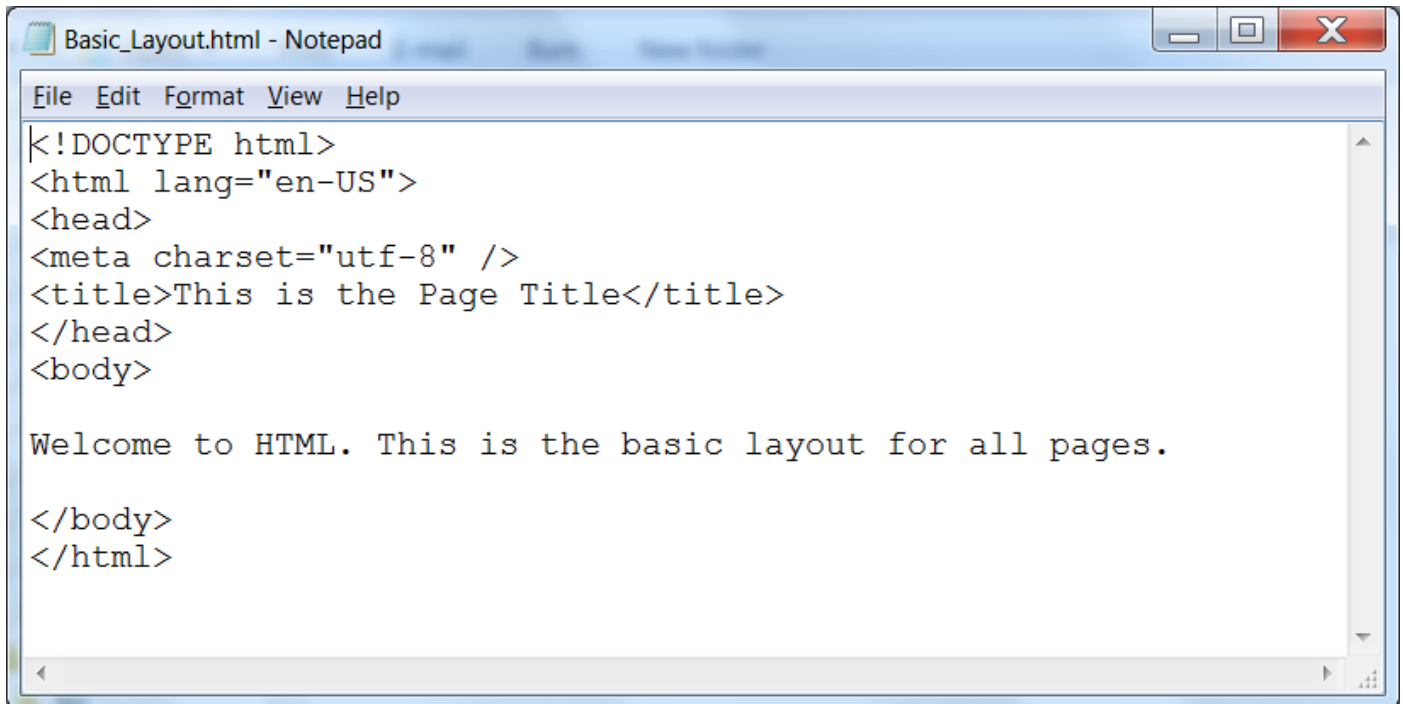
However, there are ways to open a file using other programs:

- RIGHT click on the file in Windows Explorer.
- This gives a menu of options:-
- Chose the “Open With” option:-



- This gives access to all the other potential programs that could open the selected file.
- We will initially use the Text Editor called “Notepad” to look inside our files.

Use the “Notepad “ text editor to view the file “Basic Layout.html”

A screenshot of a Notepad window titled "Basic Layout.html - Notepad". The window contains the following HTML code:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<meta charset="utf-8" />
<title>This is the Page Title</title>
</head>
<body>

Welcome to HTML. This is the basic layout for all pages.

</body>
</html>
```

Example Explained

- The **DOCTYPE** declaration defines the document type to be HTML
- The text between **<html>** and **</html>** describes an HTML document
- The `lang="en-US"` statement defines the language used
- The text between **<head>** and **</head>** provides information about the document
- The `<meta charset="utf-8" />` defines the character set that is used
- The text between **<title>** and **</title>** provides a title for the document
- The text between **<body>** and **</body>** describes the visible page content

The `<!DOCTYPE>` declaration helps the browser to display a web page correctly.

There are different document types on the web.

To display a document correctly, the browser must know both type and version.

The doctype declaration is not case sensitive. All cases are acceptable:

All HTML documents must start with a type declaration: `<!DOCTYPE html>`.

The HTML document itself begins with `<html>` and ends with `</html>`.

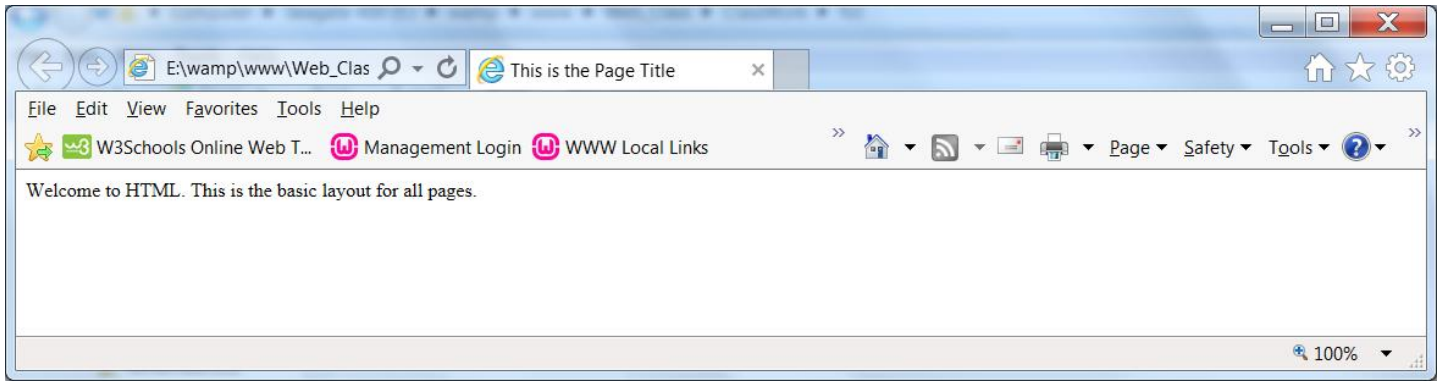
The visible part of the HTML document is between `<body>` and `</body>`.

The document language can be declared in the `<html>` tag by the **lang** attribute. `<html lang="en-US">`

Declaring a language is important for accessibility applications (screen readers) and search engines.

The first two letters specify the language (en). If there is a dialect, use two more letters (US).

Now check out the “Basic Layout.html” page in your browser.



Notice the page title to the right of the page address (URL).

The content started right at the top of the page and to the extreme left.

The font type and size were the browser default values.

M2 - How about some basic formatting?

Why do I need formatting?

Go to folder M2 and open up the file “Why_Paragraphs.html” in **Notepad** and see how the text is broken up into blocks or paragraphs. – Looks reasonable doesn’t it?

Now open up “Why_Paragraphs.html” in your **browser**. What is the problem?

Then check out “Use_Paragraphs.html” in your **browser**. What’s different and what made it so much better?

Now open “Use_Paragraphs.html” in **Notepad** and spot the difference between it and “Why_Paragraphs.html”. – Ignore the heading for now and just look at the paragraphs and note the difference that a little mark-up can make.

Now look at the mark-up around the heading. Note how it made it stand out.

Headings can come in six sizes, **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>** and **<h6>**. Check out “headings.html” to see the basic options.

Every browser has a default font size and the h1-h6 headings are set to a default proportion of that size. It is possible to change the size of any of the headings relative to the default font size by using the font-size “em” styling method. The value of the “em” is the multiplier for the font size required – “2em” yields a font twice as big as the default font size

Then check out the code and appearance of “headsized.html” for a taste of how things can be changed. We’ve started to get into styles! – More later.

How about some poetry?

Open up “Poem_Problems.html” in **Notepad**. – Looks good doesn’t it? It has all the heading and paragraph mark-ups that we saw before, so let’s now check it out in your browser: OH!

This is a problem that we can fix easily with a **Break**, one of the few mark-ups that does not have a closing mark-up. Just add **
** to the end of each line where you need to force the next bit of text to start on a new line.

Make the change and check it out.

Mark-ups we have seen so far.

Basic HTML Tags.

As a general rule, the beginning and end of any markup feature is enclosed in angle brackets < and > with a slash as the first character inside the bracket </, denoting the end. A declaration such as <!DOCTYPE html> is put at the top of the page to define the standards used.

The **H/B** column indicates whether the tag is used in the **H**ead or **B**ody section.

H/B	Begin	End	Purpose
	<html>	</html>	Encloses all HTML on a web page
	<head>	</head>	Encloses all set-up data. Must be before <body>
	<body>	</body>	Contains the detail of the visible web page
	<!--	-->	Anything in between these tags is a comment and is ignored.
H	<title>	</title>	Text that will be shown in the browser title bar
B	<h1>	</h1>	First heading. h2, h3, h4, h5, h6 define other headings
B	<p>	</p>	Encloses paragraph. Each paragraph starts on a new line under a blank line.
B	 		Breaks and starts a new line

Check out the following new mark-ups by modifying the “Poem_Problems.html” file.

H/B	Begin	End	Purpose
B			Bold text
B	<i>	</i>	Italic Text
B	<u>	</u>	Underlined text

M3 - Text Editors

The Microsoft Notepad text editor is a good basic editor that comes free with Microsoft Windows. However, in many cases there is a need to work on more than one file at a time as we shall see shortly. This can be done with Notepad, but each file is opened totally independently and this has the potential to be confusing.

As we work with increasingly more complex HTML files, keeping track of the mark-up details becomes increasingly more difficult without the use of color in the text.

There is quite a range of text editors, both free and paid, created specifically for code management. The choice of which to use can be a very personal one – Check out the following sample of free editors in the /Software/Editors folder.

Name	Color Syntax	Multiple Files	Line Numbers	Code Help	Validation	Preview	FTP
HTML-Kit	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CoffeeCup	Yes	Yes	Yes	Yes	Yes	Yes	Ext
AlleyCode	Yes	Yes	Yes	Yes	No	Yes	Ext
Editra	Yes	Yes	Yes	No	No	No	No
Programmer's Notepad	Yes	Yes	Yes	No	No	No	No
SynWrite	Yes	Yes	Yes	No	No	No	No
EditPad	No	Yes	Yes	No	No	Yes	No
NoteTab	No	Yes	No	No	No	Yes	Yes

Color Syntax: It is very important to be able to distinguish the various HTML mark-up parameters in a file. Use of color makes this much easier.

Multiple files: It is common to have more than one file open at a time and an editor that facilitates movement between files is important.

Line Numbers: These help to find a particular point in a large file.

Code Help: This can often speed up the process of writing new code.

Validation: Connects to an external program to check your code.

Preview: Quick way of viewing the web page your code produces.

FTP: **F**ile **T**ransfer **P**rotocol – The process used to transfer your code files from your local computer to the web site host.

We will use **Editra** in as the example editor in this class, but please check out the others.

M4 - HTML Styling

Every HTML element has a **default style** which is set by the browser, typically the background color is white and text color is black.

Changing the default style of an HTML element, can be done with the **style attribute**. Almost every HTML element can be styled in many appropriate ways

Styling can be added to HTML elements in 3 ways:

- Inline - using a **style attribute** in HTML elements which just styles a **single feature**.
- Internal - using a **<style> element** in the HTML **<head>** section and can be applied to **all elements on the single web page**.
- External - using one or more **external CSS files**. This method can apply consistent styling to **all web pages on a web site**

The most common way to add styling is to keep the styles in separate CSS files.

Inline HTML Styling

The Style Attribute

The HTML style attribute for in-line styling is placed in the **<body>** section in the affected markup and has the following **syntax**:

```
style="property:value;"
```

Note the two parts: “Property” and “Value”. Also note the colon (:) after the “Property” element and the semicolon (;) after the “Value”.

Several styles can be strung together as follows:-

```
style="property1:value1; property2:value2; property3:value3;"
```

Strictly speaking, the semicolon after the LAST value can be omitted, but is recommended that it be routinely included.

Now check out **Inline_Text_Styling.html**

Internal Styling (Internal CSS)

An internal style **sheet** can be used to define a common style for all HTML elements on a page.

Internal styling is defined in the **<head>** section of an HTML page, using a **<style>** element marked- up in the manner shown below:-

```
<style> element { property:value; }</style>
```

Any number of element / property / value markups can be placed between the **<style>** and **</style>** tags. Note the curly brackets surrounding the Property and Value parts and also the colon and semicolon use.

Look at **Internal_Styling.html**.

External CSS files

Using an external .CSS file makes it easy to apply consistent styles across all the pages on a web site. This process avoids having to type in the style details multiple times on multiple web pages. Obviously each web page file needs to define **what** css file it needs to link to enable the browser to apply the styles properly. This is achieved by a simple **link** mark-up in the **Head** section of the html file.

```
<link rel="stylesheet" href="css/stylesheetname.css"/>
```

The “href” part identifies the name and location of the stylesheet. It is normal practice to put stylesheets in their own .css subdirectory which makes them easier to find on large website.

The basic format of the entries on the css style sheet file is similar to the inline and internal methods:-

```
element { property1:value1; property2:value2; }
```

Examples:-

```
h1 {font-weight: bold; font-size: 100px; color: #ff0000;}
```

```
h2 {font-weight: normal; font-size: 50px; color: #000000;}
```

More about the color numbers later.

Now check out **External_css_Styling.html** and **externalcss.css**.

Where is the css file? Why?

CSS Selectors

Sometimes it is desired to change the styling within a particular element. For example, it may be required to change the styling of a complete particular paragraph or paragraphs to emphasize a particular point. This can be achieved in a couple of ways:-

- By allocating a unique **Identity (ID)** to a single individual paragraph along with styling unique to the ID.
- By giving a group of paragraphs a **class name** along with styling for that class.

The html markup for a paragraph with an **identity** would be:-

```
<P id="paratwo">
```

Note the quotes around the name. The corresponding css style markup would begin:-

```
p#paratwo {b.....}
```

Note the “#” before the identity name. The part inside the curly brackets is the normal markup for the style required. The “p” could be omitted from the HTML markup as the ID name should be unique to the one item that is to be different.

The html markup for each of the paragraphs with a **class** would be:-

```
<p class="blueback">
```

The css **class** markup is similar except that a period (.) is used instead of the #:-

```
p.blueback {bac.....}
```

Note that although the above description uses paragraph markup, the selector process can be applied to pretty much any html element.

Now check out **css_Selector_Styling.html** and **morecss.css**.

H5 - Screen Colors

Once upon a time, computer monitors and television sets had just two colors – black and white. Later, the early “color” monitors struggled to reproduce photos as they were only capable of producing 16 colors. This has changed with modern devices on which several million color shades are possible.

It has long been known that by mixing different proportions of different primary color paints, new colors could be produced. In the electronics world the three primary colors used are Red, Green and Blue or “RGB”.

Think of three flashlights, one of each of the RGB colors with the capability to control the intensity of light produced graduated from none (0) to fully on (255).

Why 0 to 255? In a 16 bit computer, one byte of data is 16 bits long and can be represented by the hexadecimal numbering system which counts as follows:-

0 1 2 3 4 5 6 7 8 9 a b c d e f

Use two bytes and you have the ability to count from 0 to 255. ($16 \times 16 = 256$ but one number is zero (0) so the highest number is 255. This is written as “ff” in hexadecimal.

The result is that with three colors each with 256 intensity levels you have the potential for $256 \times 256 \times 256 = 16777216$ colors.

Let’s check out **Stanford_Color.html** in our browser:-

All of the 3 colors are originally set at zero – no light – result = black.

Now slide the Red button all the way to the right and eventually you get bright red.

Now bring the Green button up to the red level. What Color do you get now?

Finally slide the Blue button up to the maximum value – and get – White!

Try various levels of the three colors and note the result. The bars at the sides and bottom of the color box show the proportions of each of the primary colors.

As you move the 3 sliders, note the color values and their hexadecimal equivalent

Note the “#” in front of the hexadecimal number. This will tell the computer that the number following is a hexadecimal color number.

The `Stanford_Colors.html` file lets you play with the RGB scheme, combining red, green, and blue light to make any color. The sliders control the red green and blue lights, each ranging from 0 (off) to 255 (maximum). The intersecting rectangles show the result of adding the red, green, and blue light together -- any color can be created in this way.

To make pure red, green, or blue light, just turn up that color, leaving the other two at 0. A few other common combinations:

All at max (255) = white

All at min (0) = black

red + green = yellow

red + blue = purple

green + blue turquoise

Dark yellow -- make yellow, then reduce both red and green

Orange -- make yellow, but more red, less green

Light, pastel green -- make pure green, then turn up both red and blue some equally (going towards white)

Light gray -- make white, then turn all three down a bit equally.

There are several methods commonly used to select colors for an HTML page, but let's look at a range of colors by opening **`Color_Choice.html`** in your browser.

At the bottom of the page you will see the 16 basic colors. Above is a range of possibilities.

Not the hexadecimal number for each of the colors. You can highlight the color number and use CTRL/C to copy the color number to the clipboard and CTRL/V to paste it into your HTML code.

Now look at **`Color_Choice.html`** in your editor.

The “``” element is used to encapsulate the printed hexadecimal code and print it in either black or white so that is readable on the color background.

The “`<table>`”, “`<tr>`” and “`<td>`” elements will be explored in the next module.

M6 - Images

Adding images to an HTML web page can greatly improve the “feel” of your web page. This is easily achieved by using the “img” tag in this manner:-

```

```

The critical part is the “src” part which tells the system where to go to get the image. Normal practice on a web site is to keep all images in a sub folder called “images”. This process is called **Relative Addressing** and has the advantage that you can move all the files in a web site, say from your computer to the web site host, and all of the internal links will remain valid.

The actual image files are typically JPG or GIF and could be PNG.

Many graphics files are very large and may take quite a while to load if the internet connection is slow, so there is some benefit in using a tool such as Photoshop to reduce the size of the file. These days’ multi-megapixel cameras are available and give a far greater fidelity than many typical PC monitors are capable of displaying.

The “alt” tag contains a few words that describe the picture and will display if the picture cannot be shown.

Now check out “**ImageFromPhoto.html**” in your browser and discuss the problems with what you see.

Image Sizing.

The “ImageFromPhoto.html” file is just as it came off the camera and is 850 kilobytes in size. The browser displays it pixel by pixel so only a small portion of the total image is visible in the browser’s viewport.

The viewed image can be made to fit suitably on the screen using the “Width” and “Height” tags with appropriate dimensions in pixels. However, the large file still has to be downloaded into the browser and then computer resources are then required to suitably resize the image to the required dimensions.

Check out the coding format and the result from “**Image_Sized.html**”.

A far better method, if you have access to an image editor such as Photoshop is to reduce the fidelity to an appropriate level. A useful tool for deciding what size to aim for is JRuler which is in the software section of the information for this course.

Using this approach the file size was reduced to 32 kilobytes, about 26 times smaller than the original for the same size image and fidelity as the original reduced image. See the file “**Image_Reduced.html**” which uses the smaller image file “**TBC500.jpg**”.

Images and Text.

Typically, the browser displays the information from the HTML file in the order in which it appears in the file. See an example of this in “**ImageAndText.html**”. Note that the text does not use the blank space beside the image. The view can be improved by using the “float” function which tells the browser where to move the image to facilitate text wrapping.

The result shown in “**ImageAndFloat.html**” shows the improvement produced with a “left” float. Change the float from “left” to “right” and check the result.

One thing that doesn’t look too good is the way that the edge of the text is hard against the edge of the image. You may have noticed some “margin” styling in the “**ImageAndFloat.html**” file, but all the values are set to zero. Try changing each of the values and note the resulting appearance in the browser. More about “margins” later.

M7 - Hyperlinks

The hyperlink mechanism is the feature that defines “The Web” It is the feature that allows you to go directly from one page on any website to any other page on any other site anywhere in the world.

Every web page has what amounts to a unique address, often referred to as a “URL” or “**U**niform **R**esource **L**ocator”. Taking the Calendar House web site as an example, the full address is <http://CalendarHouse.org>. A Domain Name Server links the requesting computer to the web site host and searches for a file like “**index.html**”. The “index.html” file is the web site **Home Page**. Note that the “index” part is the important bit, the second part could be php , asp or some extension other than html if alternative programming is used.

Once the home page has been reached more hyperlinks enable the user to select other pages. On the Old Calendar House site example clicking on the “Welcome” menu item would use a hyperlink to take you to the local “welcome.html” page which has a full URL of <http://www.CalendarHouse.org/welcome.html> .

The actual html code to form a hyperlink is located within an anchor tag as below:

```
<a href "LLLLL">TTTTT</a>
```

Where “LLLLL” is the URL or local file to be connected to, and
TTTTT is text to be clicked on which describes what the link is for.

Examples:

```
<a href "http://www.Southington.org">Go to the town website</a>
```

```
<a href "welcome.html">welcome</a>
```

Note that the visible text part (TTTTT in the description above) can be made to change color and/or become underlined by using suitable styling markup in the linked css file.

Check out the mini web site links by looking at “**ch_index.html**” in Module 7.

M8 - Containers – Tables and Divs

Tables

- **Tables are very useful things in Web sites. They perform several functions:**

- Present tabular information in a traditional table format

- Allow precise placement of images and text on an HTML page. To the browser viewer, it usually isn't obvious that a table is even being used

- Special purpose things like putting frames around text and pictures

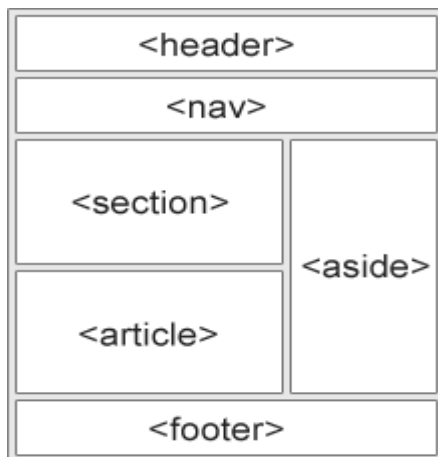
- **The table tags are:**

Tag	Description
<table> and </table>	Defines the beginning and end of the table
<caption> and </caption>	Provides a caption for a table, centered and bold
<tr> and </tr>	Defines a row in the table
<th> and </th>	Defines a bold, centered, column heading
<td> and </td>	Defines a cell in the table

Check out [sample-table.html](#).

DIVs and the development of other containers.

Earlier versions of HTML featured an element called a **div** which is basically a rectangular box with a default width of the screen which could be used to structure the layout of a web page. HTML5 recognized that there are usually distinct areas of a page for which div areas were being used and so devised a number of “divs” for specific purposes as the diagram below shows:



Header - Defines a header for a document or a section

Nav - Defines a container for navigation links

Section - Defines a section in a document

Article - Defines an independent self-contained article

Aside - Defines content aside from the content

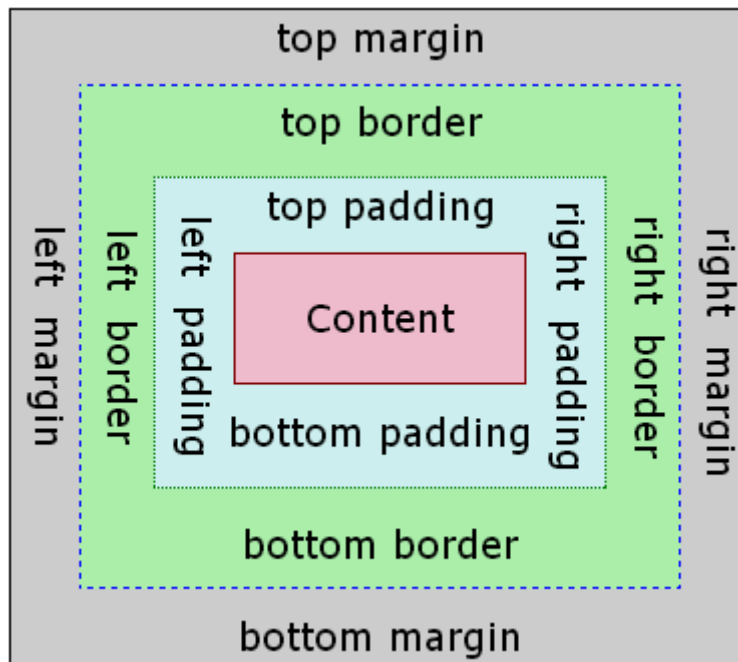
Footer - Defines a footer for a document or a section

Each of the above containers can be individually styled, sized, colored and positioned using css. This enables unique web site layouts to be achieved.

Check out **City_of_London.html** in both your browser and in the text editor. Note that the styling is included rather than in a separate css file solely to make it easier to understand the page structure.

Look at **Borders_of_London.html** to see the borders of each of the containers.

The actual content in each of the containers can be separated from the borders and the adjacent containers by setting each of the parameters shown in the following diagram:



Each of the parameters can be set separately for each of the four sides of the container if required. By default they are all zero. Examples are:-

`Padding: 5px;` = 5 pixels on each side

`Padding: 5px 9px;` = 5 pixels top and bottom, 9 pixels on each side

`Padding: 5px 9px 11px;` = 5 top, 9 left and right, 11 at the bottom

`Padding: 5px 9px 11px 7px;` = 5 top, 9 right, 11 bottom, 7 left (Clockwise)

See an example of the effects of padding, borders and margins by looking at **edgework.html**.

Note that in every case the content size is the same at 300 x 300 pixels.

Change the size of the viewable area and note the movement that results. What happens if the “**float: left;**” style is deleted?

Container Positioning

Each of the containers can be positioned in a number of ways. Without positioning or use of the “**float: left;**” or “**float: right;**” instruction each container positions its top left corner adjacent to the bottom right corner of the preceding element as based on the order of appearance in the code file, or on the next line if there is not adequate space in the viewport. This default positioning is known as “**Static**” positioning.

“**Relative**” positioning can be used to position the top left corner so many pixels right or left and up or down relative to the position it would normally be if the positioning requirement is not stated.

“**Absolute**” positioning is used to position the top left corner normally relative to the top left corner of the page.

If a page is scrolled up, down or sideways, containers positioned by any of the above 3 methods will move with the detail on the rest of the page. However, if a “**Fixed**” positioning method is used, the container if positioned relative to the browser window and does NOT move if the page is scrolled.

M9 - Putting it all together

Now that we have the basics, it's time to put it all together on a web site. By definition, a web site consists of a number of connected pages, but it's important to have a consistent "look and feel" across the whole site. This means that the general positioning of features, colors and text fonts and sizes need to be considered up front. This is where the use of css style sheets can save time as the bulk of the styling is in one place. It may also help to sketch out a simple diagram of where the various pieces go on each page.

In creating each page, it is a good idea to code the page piece by piece and test the page frequently as each section is created. Note also that all the browsers out there may render a page in a slightly different manner, so a check with some of the more popular browsers is a good idea.

A web site will have a number of internal links. It is important that all links are tested both to and from each page. External links also need to be checked out and also re-checked over time as the external site may be modified or pages may be taken down.

One problem that may occur as more pages are added to a web site is where to put all the links. A horizontal menu array is limited by the viewable screen width and the length of the wording of each link, so brevity is important. A vertical menu provides more width for the wording of the links, but as more pages are added, it may become necessary to scroll down the page to see the menu items at the bottom of the list. Most people don't like scrolling, particularly horizontally, so forward planning is essential.

It is desirable that the file name for each page is short and simple and reflects the content of the page. An exception to this is the home page for the site which is known as the **index** page and should have the name **index.html**. The reason for this is to shorten what the page visitor has to type to get to the web site – the browser will automatically look for a page named **index** and open it. So, entering www.CalendarHouse.org will automatically take you to the **index** home page. If there is no index page the browser will give an error. An alternative is to enter the fully qualified page address like www.CalendarHouse.org/welcome.html which is much more typing.

Check out the mini web site in Module 9. Which file will you open to start the site?

This site has a few minor issues, but it works. How many issues can you find?

M10 - File Transfer Protocol

Development of web pages is best carried out off line where mistakes can be made without the public seeing them. However, the intent is to eventually make the information in the pages publically accessible and this requires that the files be uploaded to a web service provider.

The web service provider will host your web site, usually for a small recurring fee. Free web hosting is available, but usually only if you allow advertizing to be displayed.

The class will be using the unused www.SouthingtonSeniornet site, but we will not be using the sites own home page as it is used to direct customers to the current Calendar House site. A link on the Calendar house site will be set up to direct users wishing to see your web sites to an internal home page on the old site where they can select which of your sites they wish to visit. This process provides a measure of security to the Calendar House site just in case mistakes are made. So, how do we get your website files to the hosting site?

A process commonly referred to as **FTP (File Transfer Protocol)** is used. We will be using an FTP program called FileZilla which we will be downloading from the web. You will need to know if the Windows operating system on the computer that you are using is 32 bit or 64 bit.

The FileZilla home page is <https://filezilla-project.org/index.php> .

See <https://wiki.filezilla-project.org/Documentation> for detailed documentation

It is recommended that you download from the

https://filezilla-project.org/download.php?show_all=1 page as this covers the two windows options as well as other operating systems.

We will cover the process of downloading and installing FileZilla in class and also the specific set-up required for you to link with the allocated space for your web site.

Once this is all set up you will find that transferring your code pages to the host site is surprisingly easy.