HTML and CSS

Intro into Web Code

This is a lesson-by-lesson walk through to doing your first web page using HTML and CSS.

Information Technology Fundamentals

HTML

The thing to keep in mind is that HTML and CSS are all about separating the content (HTML) and the presentation (CSS). HTML is nothing more than fancy structured content and the visual formatting of that content will come later when we tackle CSS.

If you have looked at other HTML tutorials, you might have found that they mention certain things that HTML Dog does not. This is because many methods are obsolete, non-standard or just plain bad practice. Getting into the frame of mind of doing things the right way from the start will turn in to much better results in the end.

Lesson 1: Getting Started

Most of the stuff on the web is no different than the stuff on your computer - it's just a whole load of files sorted into a whole load of directories.

HTML files are nothing more than simple text files, so to start writing in HTML, you need nothing more than a simple text editor.

Notepad is a common text editor (on Windows this is usually found under the Programs > Accessories menu).

Type this in to your text editor:

```
This is my first web page
```

Now create a folder called 'html' in your C drive (or anywhere else you fancy) and save the file as "myfirstpage.html". It is important that the extension ".html" is specified - some text editors, such as Notepad, will automatically save it as ".txt" otherwise.

To look at HTML files, they don't even need to be on the web. Open a web browser such as **Firefox** or **Internet Explorer** and in the address bar, where you usually type web addresses, type in the location of the file you just saved (for example, "c:\html\myfirstpage.html") and hit return. Alternatively, go to the File menu of the browser, select Open, and browse for the file.

Pow. There it is. Your first web page. How exciting. And all it took was a few typed words.

We've said here to use a basic text-editor, such as Notepad, but you may be tempted to use a dedicated software program such as **Macromedia Dreamweaver** or **Microsoft Frontpage**.

You should be very careful when using these programs, especially if you are a beginner, because they often throw in unnecessary or non-standard code to "help" you.

If you're serious about learning HTML, you should read through a tutorial such as this first, so that you at least have a basic understanding of what is going on.

Software programs such as these will never give you the same control over a web page as coding by hand.

Lesson 2: Tags, Attributes, and Elements

Although the basics of HTML are plain text, we need a bit more to make it a valid HTML document.

Tags

The basic structure of an HTML document includes **tags**, which surround content and apply meaning to it.

Change your document so that it looks like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<body>
                    This is my first web page
</body>
</html>
```

Now save the document again, go back to the web browser and select "refresh" (which will reload the page).

The appearance of the page will not have changed at all, but the purpose of HTML is to apply meaning, not presentation, and this example has now defined some fundamental elements of a web page.

The first line on the top that starts <!DOCTYPE... is to let the browser know that you know what the hell you're doing. You may think that you don't actually know what you're doing yet, but it's important to stick this in. If you don't, browsers will switch into "quirks mode" and act in a very peculiar way. Don't worry about this just yet, you can learn more about "document types" in the HTML Advanced Tutorial if you really want to. For the moment, just remember to shove this line at the top of your web pages and you're laughin'.

To get back to the point, <html> is the **opening tag** that kicks things off and tells the browser that everything between that and the </html> **closing tag** is an HTML document. The stuff between <body> and </body> is the main content of the document that will appear in the browser window.

Closing tags

The </body> and </html> close their respective tags. ALL HTML tags should be closed. Although older versions of HTML lazily allowed some tags not to be closed, latest standards require all tags to be closed. This is a good habit to get into anyway.

Not all tags have closing tags like this (<html></html>) some tags, which do not wrap around content will close themselves. The line-break tag for example, looks like this :
. We will come across these

examples later. All you need to remember is that all tags must be closed and most (those with content between them) are in the format of opening tag → content → closing tag.

Attributes

Tags can also have **attributes**, which are extra bits of information. Attributes appear inside the opening tag and their value is always inside quotation marks. They look something like <tag attribute="value">Margarine</tag>. We will come across tags with attributes later.

Elements

Tags tend not to do much more than mark the beginning and end of an **element**. Elements are the bits that make up web pages. You would say, for example, that everything that is in-between and includes the <body> and </body> tags is the body element. As another example, whereas '<title>' and '</title>' are *tags*, '<title>Rumple Stiltskin</title>' is a title *element*.

Lesson 3: Page Titles

All HTML pages should have a page **title**.

To add a title to your page, change your code so that it looks like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>
                    <title>My first web page</title>
</head>

<body>
                    This is my first web page
</body>

</html>
```

We have added two new elements here, that start with the **head** tag and the **title** tag (and see how both of these close).

The head element (that which starts with the <head> opening tag and ends with the </head> tag) appears before the body element (starting with <body> and ending with </body>) and contains information *about* the page. The information in the head element does not appear in the browser window.

We will see later on that other elements can appear inside the head element, but the most important of them is the **title** element.

If you look at this document in the browser (save and refresh as before), you will see that "My first web page" will appear on the title bar of the window (not the actual canvas area). The text that you put in between the title tags has become the title of the document (surprise!). If you were to add this page to

your 'favourites' (or 'bookmarks', depending on your browser), you would see that the title is also used there.

Lesson 4: Paragraphs

Now that you have the basic structure of an HTML document, you can mess about with the content a bit.

Go back to your text editor and add another line to your page:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>
                        <title>My first web page</title>
</head>

<body>
                        This is my first web page
                        How exciting
</body>

</html>
```

Look at the document in your browser.

You might have expected your document to appear as you typed it, on two lines, but instead you should see something like:

This is my first web page How exciting.

This is because web browsers don't usually take any notice of what line your code is on. It also doesn't take any notice of spaces (you would get the same result if you typed "This is my first web page How exciting").

If you want text to appear on different lines, you need to explicitly state that.

Change your two lines of content so that they look like this:

```
<p>This is my first web page</p>
<p>How exciting</p>
```

The **p** tag is for **paragraph**.

Look at the results of this. The two lines will now appear on two lines.

Think of the HTML content as if it were a book - with paragraphs where appropriate.

Emphasis

You can emphasise text in a paragraph using em (emphasis) and strong (strong emphasis). These are two ways of doing pretty much the same thing, although traditionally, browsers display em in italics and strong in bold.

```
<p>Yes, that <em>is</em> what I said. How <strong>very</strong> exciting.</p>
```

Line breaks

The line-break tag can also be used to separate lines like this:

```
This is my first web page<br />
How exciting
```

However, this method is over-used and shouldn't be used if two blocks of text are intended to be separate from one another (because if that's what you want to do you probably want the p tag).

Note that because there's no content involved with the line-break tag, there is no closing tag and it closes itself with a "/" after the "br".

Lesson 5: Headings

The p tag is just the start of text formatting.

If you have documents with genuine **headings**, then there are HTML tags specifically designed just for them.

They are **h1**, **h2**, **h3**, **h4**, **h5** and **h6**, h1 being the almighty emperor of headings and h6 being the lowest pleb.

Change your code to the following:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>
                <title>My first web page</title>
</head>

<body>
                <h1>My first web page</h1>

                <h2>What this is</h2>
                <p>A simple page put together using HTML</p>

                <h2>Why this is</h2>
                <p>To learn HTML</p>
</body>
```

```
</html>
```

Note that the h1 tag is only used once - it is supposed to be the main heading of the page and shouldn't be used multiple times.

h2 to h6 however, can be used as often as you desire, but they should always be used in order, as they were intended. For example, an h4 should be a sub-heading of an h3, which should be a sub-heading of an h2.

Lesson 6: Lists

There are three types of list; unordered lists, ordered lists and definition lists. We will look at the first two here, and definition lists in the HTML Intermediate Tutorial.
Unordered lists and ordered lists work the same way, except that the former is used for non-sequential lists with list items usually preceded by bullets and the latter is for sequential lists, which are normally represented by incremental numbers.

The ul tag is used to define unordered lists and the ol tag is used to define ordered lists. Inside the lists, the li tag is used to define each list item.

Change your code to the following:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>
	<title>My first web page</title>
</head>

<body>
	<h1>My first web page</h1>

	<h2>What this is</h2>
	<p>A simple page put together using HTML</p>

	<h2>Why this is</h2>
	<ul>
		<li>To learn HTML</li>
		<li>To show off</li>
		<li>Because I've fallen in love with my computer and want to give her some HTML
loving.</li>
	</ul>

</body>

</html>
```

If you look at this in your browser, you will see a bulleted list. Simply change the ul tags to ol and you will see that the list will become numbered.

Lists can also be included in lists to form a structured hierarchy of items.

Replace the above list code with the following:

```
<ul>
        <li>To learn HTML</li>
        <li>
                To show off
                <ol>
                        <li>To my boss</li>
                        <li>To my friends</li>
                        <li>To my cat</li>
                        <li>To the little talking duck in my brain</li>
                </ol>
        </li>
        <li>Because I've fallen in love with my computer and want to give her some HTML loving.</li>
</ul>
```

Et voil&#65533;. A list within a list. And you could put another list within that. And another within that. And so on and so forth.

Lesson 7: Links

So far you've been making a stand-alone web page, which is all very well and nice, but what makes the internet so special is that it all links together.

The 'H' and 'T' in 'HTML' stand for 'hypertext', which basically means a system of linked text.

An anchor tag (a) is used to define a link, but you also need to add something to the anchor tag - the destination of the link.

Add this to your document:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>
        <title>My first web page</title>
</head>
```

```
<body>

	<h1>My first web page</h1>

	<h2>What this is</h2>
	<p>A simple page put together using HTML</p>

	<h2>Why this is</h2>
	<p>To learn HTML</p>

	<h2>Where to find the tutorial</h2>
	<p><a href="http://www.htmldog.com">HTML Dog</a></p>

</body>

</html>
```

The destination of the link is defined in the href attribute of the tag. The link can be absolute, such as "http://www.htmldog.com", or it can be relative to the current page.

So if, for example, you had another file called "flyingmoss.html" then the line of code would simply be The miracle of moss in flight or something like this.

A link does not have to link to another HTML file, it can link to any file anywhere on the web.

A link can also send a user to another part of the same page they are on. You can add an id attribute to just about any tag, for example <h2 id="moss">Moss</h2>, and then link to it by using something like this: Go to moss. Selecting this link will scroll the page straight to the element with that id.

The a tag allows you to open the link in a newly spawned window, rather than replacing the web page the user is on, which at first thought may sound like a good idea as it doesn't take the user away from your site.

There are a number of reasons why you shouldn't do this however.

From a usability point of view, this method breaks navigation. The most commonly used navigation tool on a browser is the "back" button. Opening a new window disables this function.

On a wider, more general usability point, users do not want new windows to be popping up all over the place. If they want to open a link in a new window then they can choose to do so themselves.

Lesson 8: Images

Things might seem a little bland and boring with all of this text formatting. Of course, the web is not just about text, it is multi-media and the most common form of media is the image.

The img tag is used to put an image in an HTML document and it looks like this:

```
<img src="http://www.htmldog.com/images/logo.gif" width="157" height="70" alt="HTML Dog logo" />
```

The src attribute tells the browser where to find the image. Like the a tag, this can be absolute, as the above example demonstrates, but is usually relative. For example, if you create your own image and save it as "alienpie.jpg" in a directory called "images" then the code would be <img src="images/alienpie.jpg"...

The width and height attributes are necessary because if they are excluded, the browser will tend to calculate the size as the image loads, instead of when the page loads, which means that the layout of the document may jump around while the page is loading.

The alt attribute is the alternative description. This is used for people who cannot or choose not to view images. This is a requirement in the latest versions of HTML.

Note that, like the br tag, because the img tag does not have a closing tag, it closes itself, ending with "/>"

The construction of images for the web is a little outside of the remit of this website, but it is worth noting a few things...

The most commonly used file formats used for images are GIFs and JPEGs. They are both compressed formats, and have very different uses.

GIFs can have no more than 256 colours, but they maintain the colours of the original image. The lower the number of colours you have in the image, the lower the file size will be.

GIFS SHOULD BE USED FOR IMAGES WITH SOLID COLOURS.

JPEGs on the other hand use a mathematical algorithm to compress the image and will distort the original slightly. The lower the compression, the higher the file size, but the clearer the image.

JPEGS SHOULD BE USED FOR IMAGES SUCH AS PHOTOGRAPHS.

Images are perhaps the largest files a new web designer will be handling. It is a common mistake to be oblivious to the file size of images, which can be extremely large. Web pages should download as quickly as possible, and if you keep in mind that most people still use modems that download at less than 7Kb a second (realistically it is less than 5Kb), you can see how a large file will greatly slow down the download time of a full page.

You need to strike a balance between image quality and image size. Most modern image manipulation programs allow you to compress images and the best way to figure out what is best suited for yourself is trial and error.

Lesson 9: Tables

Across the worldwide web, HTML tables are used and abused to layout pages. We will come across how to layout a page without tables, in the CSS Advanced Tutorial. The correct use for tables is to do exactly what you would expect a table to do - to structure tabular data.

There are a number of tags used in tables, and to fully get to grips with how they work is probably the most difficult area of this HTML Beginners Tutorial.

Copy the following code into the body of your document and then we will go through what each tag is doing:

```
<table>
	<tr>
		<td>Row 1, cell 1</td>
		<td>Row 1, cell 2</td>
		<td>Row 1, cell 3</td>
	</tr>
	<tr>
		<td>Row 2, cell 1</td>
		<td>Row 2, cell 2</td>
		<td>Row 2, cell 3</td>
	</tr>
	<tr>
		<td>Row 3, cell 1</td>
		<td>Row 3, cell 2</td>
		<td>Row 3, cell 3</td>
	</tr>
	<tr>
		<td>Row 4, cell 1</td>
		<td>Row 4, cell 2</td>
		<td>Row 4, cell 3</td>
	</tr>
</table>
```

The table element defines the table.

The tr element defines a table row.

The td element defines a data cell. These must be enclosed in tr tags, as shown above.

If you imagine a 3x4 table, which is 12 cells, there should be four tr elements to define the rows and three td elements within each of the rows, making a total of 12 td elements.

Lesson 10: Forms

Forms can be used to send data across the web and are often used as contact forms to convert information inputted by a user into an email, such as the one used on this website.

On their own, forms are useless. They need to be hooked up to a program that will process the data inputted by the user. These take all manner of guises and are outside of the remit of this website. If you use an internet service provider to host your HTML, they will be able to help you with this and will probably have clear and simple instructions on how, for example, to make a form-to-email form work.

The basic tags used in the actual HTML of forms are form, input, textarea, select and option.

form defines the form and within this tag, there is one required action attribute which tells the form where its contents will be sent to when it is submitted.

The optional method attribute tells the form how the data in it is going to be sent and it can have the value get (which is default) or post. This is commonly used, and often set to post which hides the information (get latches the information onto the URL).

So a form element will look something like this:

```
<form action="processingscript.php" method="post">

</form>
```

The input tag is the daddy of the form world. It can take ten forms, outlined below:

- <input type="text" /> is a standard textbox. This can also have a value attribute, which sets the initial text in the textbox.
- <input type="password" /> is similar to the textbox, but the characters typed in by the user will be hidden.
- <input type="checkbox" /> is a checkbox, which can be toggled on and off by the user. This can also have a checked attribute, which would be used in the format <input type="checkbox" checked="checked" />, and makes the initial state of the check box to be switched on, as it were.
- <input type="radio" /> is similar to a checkbox, but the user can only select one radio button in a group. This can also have a checked attribute, used in the same way as the checkbox.
- <input type="file" /> is an area that shows the files on your computer, like you see when you open or save a document in most programs, and is used to enable users to upload files.
- <input type="submit" /> is a button that when selected will submit the form. You can control the text that appears on the submit button (as you can with button and reset types - see below) with the value attribute, for example <input type="submit" value="Ooo. Look. Text on a button. Wow" />.
- <input type="image" /> is an image that will submit the coordinates of where the user clicked on it. This also requires a src attribute, like the img tag.
- <input type="button" /> is a button that will not do anything without extra code added.
- <input type="reset" /> is a button that when selected will reset the form fields to their default values.
- <input type="hidden" /> is a field that will not be displayed and is used to pass information such as the page name that the user is on or the email address that the form should be posted to.

Note that the input tag closes itself with a "/>" at the end.

A textarea is, basically, a large textbox. It requires a rows and cols attribute and is used like this:

```
<textarea rows="5" cols="20">A big load of text here</textarea>
```

The select tag works with the option tag to make drop-down select boxes.

They work like this:

```
<select>
	<option value="first option">Option 1</option>
	<option value="second option">Option 2</option>
	<option value="third option">Option 3</option>
</select>
```

When the form is submitted, the value of the selected option will be sent.

Similar to the checked attribute of checkboxes and radio buttons, an option tag can also have a selected attribute, which would be used in the format <option value="mouse" selected="selected">Rodent</option>.

All of the tags mentioned above will look very nice presented on the page, but if you hook up your form to a form-handling program, they will all be ignored. This is because the form fields need names. So to all of the fields, the attribute name needs to be added, for example <input type="text" name="talkingsponge" />

A form might look like the one below. (Note: this form will not work unless there is a "contactus.php" file, which is stated in the action attribute of the form tag, to handle the submitted date)

```
<form action="contactus.php" method="post">

	<p>Name:</p>
	<p><input type="text" name="name" value="Your name" /></p>

	<p>Comments: </p>
	<p><textarea name="comments" rows="5" cols="20">Your comments</textarea></p>

	<p>Are you:</p>
	<p><input type="radio" name="areyou" value="male" /> Male</p>
	<p><input type="radio" name="areyou" value="female" /> Female</p>
	<p><input type="radio" name="areyou" value="hermaphrodite" /> An  hermaphrodite</p>
	<p><input type="radio" name="areyou" value="asexual" checked="checked" /> Asexual</p>

	<p><input type="submit" /></p>

	<p><input type="reset" /></p>

</form>
```

There is a whole other level of complexity you can delve into in the HTML Advanced Tutorial if you are so inclined.

Lesson 11: Putting It All Together

If you have gone through all of the pages in this HTML Beginner Tutorial then you should be a competent HTMLer.

In fact, due to the fact that most people who use HTML use it rather badly, you should be better than most.

The following code incorporates all of the methods that have been explained in the previous pages:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

	<title>My first web page</title>

	<!-- By the way, this is a comment -->

</head>

<body>

<h1>My first web page</h1>

<h2>What this is</h2>
<p>A simple page put together using HTML. <strong>A simple page put together using HTML.</strong> A simple page put together using HTML. A simple page put together using HTML. A simple page put together using HTML. A simple page put together using HTML. A simple page put together using HTML. A simple page put together using HTML. A simple page put together using HTML.</p>

<h2>Why this is</h2>
<ul>
	<li>To learn HTML</li>
	<li>
		To show off
		<ol>
			<li>To my boss</li>
			<li>To my friends</li>
			<li>To my cat</li>
			<li>To the little talking duck in my brain</li>
		</ol>
	</li>
	<li>Because I've fallen in love with my computer and want to give her some HTML loving.</li>
</ul>
```

```
<h2>Where to find the tutorial</h2>
<p><a href="http://www.htmldog.com"><img src="http://www.htmldog.com/images/logo.gif"
width="157" height="70" alt="HTML Dog logo" /></a></p>

<h3>Some random table</h3>
<table border="1">
	<tr>
		<td>Row 1, cell 1</td>
		<td>Row 1, cell 2</td>
		<td>Row 1, cell 3</td>
	</tr>
	<tr>
		<td>Row 2, cell 1</td>
		<td>Row 2, cell 2</td>
		<td>Row 2, cell 3</td>
	</tr>
	<tr>
		<td>Row 3, cell 1</td>
		<td>Row 3, cell 2</td>
		<td>Row 3, cell 3</td>
	</tr>
	<tr>
		<td>Row 4, cell 1</td>
		<td>Row 4, cell 2</td>
		<td>Row 4, cell 3</td>
	</tr>
</table>

<h3>Some random form</h3>
<p><strong>Note:</strong> It looks the part, but won't do a damned thing</p>

<form action="somescript.php" method="post">

<p>Name:</p>
<p><input type="text" name="name" value="Your name" /></p>

<p>Comments: </p>
<p><textarea rows="10" cols="20" name="comments">Your comments</textarea></p>

<p>Are you:</p>
<p><input type="radio" name="areyou" value="male" /> Male</p>
<p><input type="radio" name="areyou" value="female" /> Female</p>
<p><input type="radio" name="areyou" value="hermaphrodite" /> An hermaphrodite</p>
<p><input type="radio" name="areyou" value="asexual" checked="checked" /> Asexual</p>

<p><input type="submit" /></p>

<p><input type="reset" /></p>

</form>
```

```
</body>

</html>
```

There you have it. Save the file and play around with it - this is the best way to understand how everything works. Go on. Tinker.

CSS

CSS, or **Cascading Styles Sheets**, is a way to style HTML. Whereas the HTML is the **content**, the style sheet is the **presentation** of that document.

Styles don't smell or taste anything like HTML, they have a format of '**property: value**' and most properties can be applied to most HTML tags.

Lesson 12: Applying CSS

There are three ways to apply CSS to HTML.

In-line styles are plonked straight into the HTML tags using the style attribute.

They look something like this:

```
<p style="color: red">text</p>
```

This will make that specific paragraph red.

But, if you remember, the best-practice approach is that the HTML should be a stand-alone, presentation free document, and so in-line styles should be avoided wherever possible.
Internal

Embedded, or internal styles are used for the whole page. Inside the head tags, the style tags surround all of the styles for the page.

This would look something like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>CSS Example</title>
<style type="text/css">
        p {
                color: red;
        }
```

```
        a {
                color: blue;
        }
</style>
```
...

This will make all of the paragraphs in the page red and all of the links blue.

Similarly to the in-line styles, you should keep the HTML and the CSS files separate, and so we are left with our saviour...
External

External styles are used for the whole, multiple-page website. There is a separate CSS file, which will simply look something like:

```
p {
        color: red;
}

a {
        color: blue;
}
```

If this file is saved as "web.css" then it can be linked to in the HTML like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
        <title>CSS Example</title>
        <link rel="stylesheet" type="text/css" href="web.css" />
```
...

In the CSS Advanced Tutorial, we will see that there are other ways of linking external style sheets, but this will suffice for now.

To get the most from this guide, it would be a good idea to try out the code as we go along, so start a fresh new file with your text-editor and save the blank document as "web.css" in the same directory as your HTML file.

Now change your HTML file so that it starts something like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>
```

```
<head>
	<title>My first web page</title>
	<link rel="stylesheet" type="text/css" href="web.css" />
</head>
```
...

Save the HTML file. This now links to the CSS file, which is empty at the moment, so won't change a thing. As you work your way through the CSS Beginner Tutorial, you will be able to add to and change the CSS file and see the results by simply refreshing the browser window that has the HTML file in it, as we did before.

Lesson 13: CSS Selectors, Properties, and Values

Whereas HTML has tags, CSS has 'selectors'. Selectors are the names given to styles in internal and external style sheets. In this CSS Beginner Tutorial we will be concentrating on HTML selectors, which are simply the names of HTML tags and are used to change the style of a specific tag.

For each selector there are 'properties' inside curly brackets, which simply take the form of words such as color, font-weight or background-color.

A value is given to the property following a colon (NOT an 'equals' sign) and semi-colons separate the properties.

```
body {
	font-size: 0.8em;
	color: navy;
}
```

This will apply the given values to the font-size and color properties to the body selector.

So basically, when this is applied to an HTML document, text between the body tags (which is the content of the whole window) will be 0.8 ems in size and navy in colour.
Lengths and Percentages

There are many property-specific units for values used in CSS, but there are some general units that are used in a number of properties and it is worth familiarising yourself with these before continuing.

em (such as font-size: 2em) is the unit for the calculated size of a font. So "2em", for example, is two times the current font size.

px (such as font-size: 12px) is the unit for pixels.

pt (such as font-size: 12pt) is the unit for points.

% (such as font-size: 80%) is the unit for... wait for it... percentages.

Other units include pc (picas), cm (centimetres), mm (millimetres) and in (inches).

When a value is zero, you do not need to state a unit. For example, if you wanted to specify no border, it would be border: 0.

A web page is not a static, absolute medium. It is meant to be flexible and the user should be allowed to view the web page how the hell they like, which includes the font size and the size of the screen.

Because of this, it is generally accepted that 'em' or '%' are the best units to use for font-sizes (and possibly even heights and widths, which we shall come across in the CSS Advanced Tutorial), rather than 'px', which leads to non-resizable text in most browsers, and should be used sparingly, for border sizes for example.

Colours

CSS brings 16,777,216 colours to your disposal. They can take the form of a name, an rgb (red/green/blue) value or a hex code.

red
Is the same as
rgb(255,0,0)
Which is the same as
rgb(100%,0%,0%)
Which is the same as
#ff0000
Which is the same as
#f00

There are 17 valid predefined colour names. They are aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, and yellow.

transparent is also a valid value.

The three values in the rbg value are from 0 to 255, 0 being the lowest level (for example no red), 255 being the highest level (for example full red). These values can also be a percentage.

Hexadecimal (previously and more accurately known as 'sexadecimal') is a base-16 number system. We are generally used to the decimal number system (base-10, from 0 to 9), but hexadecimal has 16 digits, from 0 to f.

The hex number is prefixed with a hash character (#) and can be three or six digits in length. Basically, the three-digit version is a compressed version of the six-digit (#f00 becomes #ff0000, #c96 becomes #cc9966 etc.). The three-digit version is easier to decipher (the first digit, like the first value in rgb, is red, the second green and the third blue) but the six-digit version gives you more control over the exact colour.

'color' and 'background-color'

Colours can be applied by using color and background-color (note that this must be the American English 'color' and not 'colour').

A blue background and yellow text could look like this:

```
h1 {
        color: yellow;
        background-color: blue;
}
```

These colours might be a little too harsh, so you could change the code of your CSS file for slightly different shades:

```
body {
        font-size: 0.8em;
        color: navy;
}

h1 {
        color: #ffc;
        background-color: #009;
}
```

Save the CSS file and refresh your browser. You will see the colours of the first heading (the h1 element) have changed to yellow and blue.

You can apply the color and background-color properties to most HTML elements, including body, which will change the colours of the page and everything in it.

Lesson 14: Text

You can alter the size and shape of the text on a web page with a range of properties, outlined below:

font-family

This is the font itself, such as Times New Roman, Arial, or Verdana.

The font you specify must be on the user's computer, so there is little point in using obscure fonts. There are a select few 'safe' fonts (the most commonly used are arial, verdana and times new roman), but you can specify more than one font, separated by commas. The purpose of this is that if the user does not have the first font you specify, the browser will go through the list until it finds one it does have. This is useful because different computers sometimes have different fonts installed. So font-family: arial, helvetica, for example, is used so that similar fonts are used on PC (which traditionally has arial, but not helvetica) and Apple Mac (which, traditionally, does not have arial and so helvetica, which it does normally have, will be used).

Note: if the name of a font is more than one word, it should be put in quotation marks, such as font-family: "Times New Roman".

font-size

The size of the font. Be careful with this - text such as headings should not just be a paragraph in a large font; you should still use headings (h1, h2 etc.) even though, in practice, you could make the font-size of a paragraph larger than that of a heading (not recommended for sensible people).

font-weight

This states whether the text is bold or not. In practice this usually only works as font-weight: bold or font-weight: normal. In theory it can also be bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800 or 900, but seeing as many browsers shake their heads and say "I don't think so", it's safer to stick with bold and normal.

font-style

This states whether the text is italic or not. It can be font-style: italic or font-style: normal.

text-decoration

This states whether the text is underlined or not. This can be:

text-decoration: overline, which places a line above the text.
text-decoration: line-through, strike-through, which puts a line through the text.
text-decoration: underline should only be used for links because users generally expect underlined

text to be links.

This property is usually used to decorate links, such as specifying no underline with text-decoration: none.

text-transform

This will change the case of the text.

text-transform: capitalize turns the first letter of every word into uppercase.
text-transform: uppercase turns everything into uppercase.
text-transform: lowercase turns everything into lowercase.
text-transform: none I'll leave for you to work out.

```
body {
	font-family: arial, helvetica, sans-serif;
	font-size: 0.8em;
}

h1 {
	font-size: 2em;
}

h2 {
	font-size: 1.5em;
}
```

```
a {
	text-decoration: none;
}

strong {
	font-style: italic;
	text-transform: uppercase;
}
```

Text spacing

The letter-spacing and word-spacing properties are for spacing between letters or words. The value can be a length or normal.

The line-height property sets the height of the lines in an element, such as a paragraph, without adjusting the size of the font. It can be a number (which specifies a multiple of the font size, so '2' will be two times the font size, for example), a length, a percentage or normal.

The text-align property will align the text inside an element to left, right, center or justify.

The text-indent property will indent the first line of a paragraph, for example, to a given length or percentage. This is a style traditionally used in print, but rarely in digital media such as the web.

```
p {
	letter-spacing: 0.5em;
	word-spacing: 2em;
	line-height: 1.5;
	text-align: center;
}
```

Lesson 15: Margins and Padding

margin and padding are the two most commonly used properties for spacing-out elements. A margin is the space outside of the element, whereas padding is the space inside the element.

Change the CSS code for h2 to the following:

```
h2 {
	font-size: 1.5em;
	background-color: #ccc;
	margin: 1em;
	padding: 3em;
}
```

You will see that this leaves one-character width space around the secondary header and the header itself is fat from the three-character width padding.

The four sides of an element can also be set individually. margin-top, margin-right, margin-bottom, margin-left, padding-top, padding-right, padding-bottom and padding-left are the self-explanatory properties you can use.

The Box Model

Margins, padding and borders (see next page) are all part of what's known as the Box Model. The Box Model works like this: in the middle you have the content area (let's say an image), surrounding that you have the padding, surrounding that you have the border and surrounding that you have the margin. It can be visually represented like this:

You don't have to use all of these, but it can be helpful to remember that the box model can be applied to every element on the page, and that's a powerful thing!

Lesson 16: CSS Borders

Borders can be applied to most HTML elements within the body.

To make a border around an element, all you need is border-style. The values can be solid, dotted, dashed, double, groove, ridge, inset and outset.

border-width sets the width of the border, which is usually in pixels. There are also properties for border-top-width, border-right-width, border-bottom-width and border-left-width.

Finally, border-color sets the colour.

Add the following code to the CSS file:

```
h2 {
        border-style: dashed;
        border-width: 3px;
        border-left-width: 10px;
        border-right-width: 10px;
        border-color: red;
}
```

This will make a red dashed border around all HTML secondary headers (the h2 element) that is 3 pixels wide on the top and bottom and 10 pixels wide on the left and right (these having over-ridden the 3 pixel wide width of the entire border).

Lesson 17: CSS Beginner Tutorial - Putting It All Together

You should already have an HTML file like the one made at the end of the HTML Beginner Tutorial, with a line that we added at the start of this CSS Beginner Tutorial, linking the HTML file to the CSS file.

The code below covers all of the CSS methods in this section. If you save this as your CSS file and look at the HTML file then you should now understand what each CSS property does and how to apply them. The best way to fully understand all of this is to mess around with the HTML and the CSS files and see what happens when you change things.

```
body {
	font-family: arial, helvetica, sans-serif;
	font-size: 80%;
	color: black;
	background-color: #ffc;
	margin: 1em;
	padding: 0;
}

/* By the way, this is a comment */

p {
	line-height: 1.5em;
}

h1 {
	color: #ffc;
	background-color: #900;
	font-size: 2em;
	margin: 0;
	margin-bottom: 0.5em;
	padding: 0.25em;
	font-style: italic;
	text-align: center;
	letter-spacing: 0.5em;
	border-bottom-style: solid;
	border-bottom-width: 0.5em;
	border-bottom-color: #c00;
}

h2 {
	color: white;
	background-color: #090;
	font-size: 1.5em;
```

```
        margin: 0;
        padding: 0.1em;
        padding-left: 1em;
}

h3 {
        color: #999;
        font-size: 1.25em;
}

img {
        border-style: dashed;
        border-width: 2px;
        border-color: #ccc;
}

a {
        text-decoration: none;
}

strong {
        font-style: italic;
        text-transform: uppercase;
}

li {
        color: #900;
        font-style: italic;
}

table {
        background-color: #ccc;
}
```