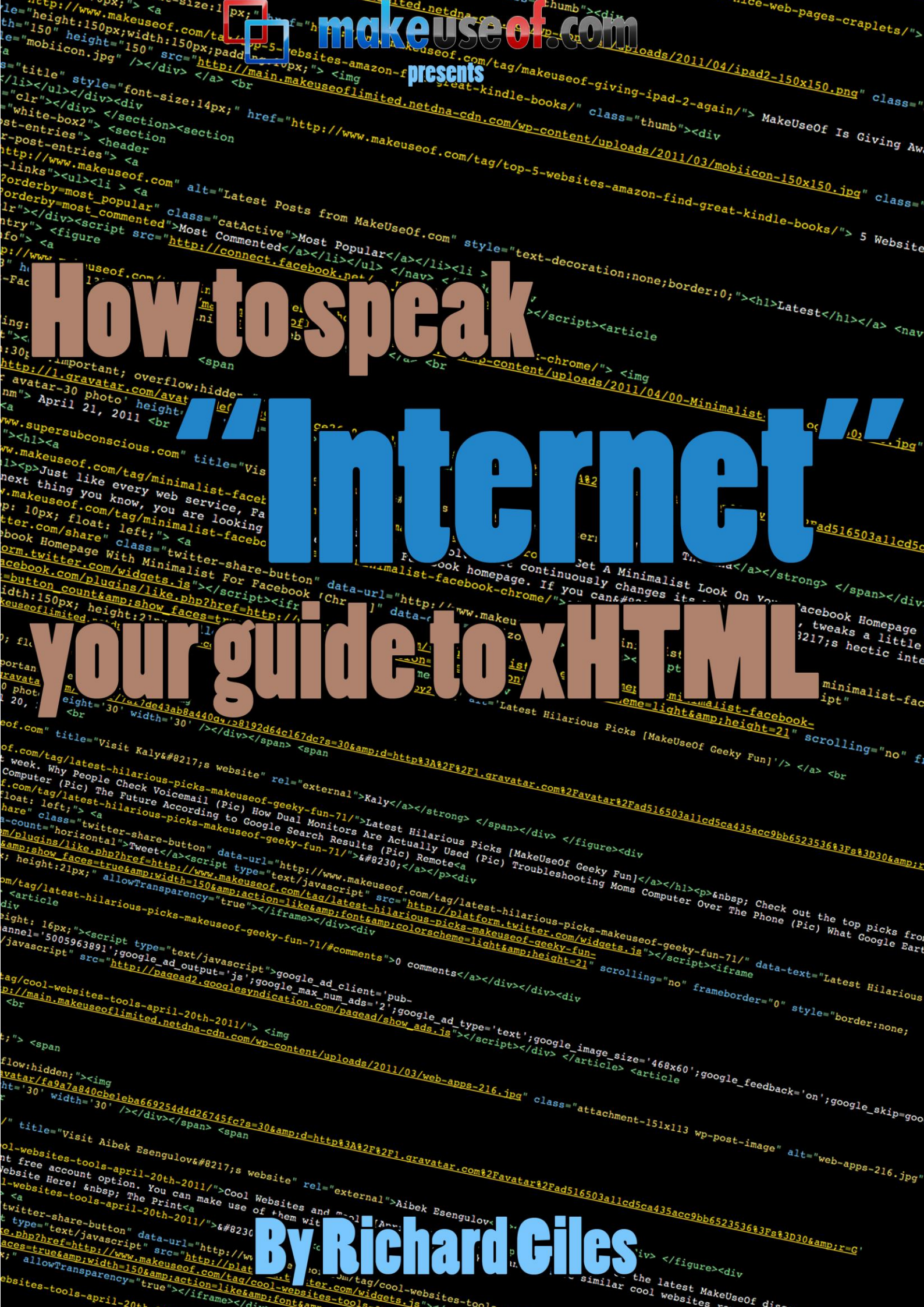
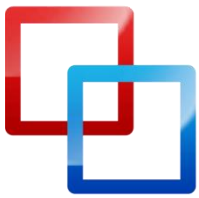

makeuseof.com
presents
How to speak
"Internet"
your guide to xHTML
By Richard Giles



makeuseof.com

LEARN TO SPEAK "INTERNET": GUIDE TO XHTML

By: Richard Giles

Edited by: Justin Pot

Table of Contents

1 – Introduction to xHTML

Welcome to the world of XHTML – **Extensible Hypertext Markup Language** – a markup language (similar to programming) that allows anyone to construct web pages with many different functions. In many ways, it's the primary language of the Internet.

So, why do we care?

Well, haven't you ever wanted to have your own website? Or make your own game? The role of this guide is to give you a taste of this powerful world. If you have any previous programming experience then you will find this easier, of course, than if you are just starting your programming adventure. Either way, I hope to explain this so even a novice can understand.

We care about xHTML because it is a strong starting point to learning the basic building blocks of the web. Social networking sites like Facebook, MySpace and Twitter use another (server-side) programming language called PHP, but it's a good idea to understand the basics before you dive headfirst into the programming world. This guide is about the basics.

If you want to know more about how the Internet works or perhaps how computer networks work with all of this technical stuff or even just how computers can be built then try these great guides from your friends at MakeUseOf:

http://www.makeuseof.com/pages/the-guide-build-your-own-pc
http://www.makeuseof.com/pages/easy-guide-computer-networks
http://www.makeuseof.com/pages/guide-file-sharing-networks
http://www.makeuseof.com/pages/download-the-ultimate-windows-7-guide
http://www.makeuseof.com/pages/download-how-the-internet-works

2 – Getting Started with xHTML

In this chapter you will learn how to create and customize websites in many different ways including learning how to:

- Add images to web pages.
- Create and use hyperlinks to navigate web pages.
- Set up lists of information using dot points and such.
- Create tables with rows and columns of random data and be able to control the formatting of said tables.

- Create and use forms that you can actually have some interaction with.
- Make web pages accessible to search engines.

All of this will be done with xHTML. Don't believe it? Read on. You would be surprised how much you can learn from such a short guide.

Before we actually get into the "coding" part of this guide, you will need some software to use so that you can edit, test and basically all around develop your programs. Go to **www.dreamspark.com** and get one of the following programs for FREE, assuming you're a student:

- Microsoft Visual Studio 2010
- Expression Studio 4

If you're not a student, you can also use **Notepad++,** which you can quite easily get from **www.notepad-plus-plus.org**

Once you get one of the programs and install it, then you can start your xHTML experience.

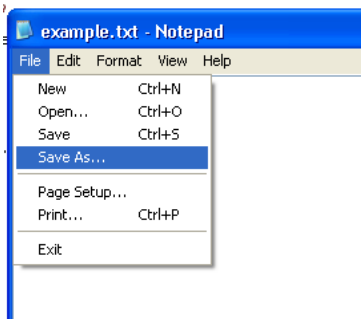
You might be using a Mac or Linux instead of Windows; you'll need to find a text editor that works for you in that case. Try to find one that shows you your line count and colours code for you.

- http://www.makeuseof.com/tag/the-top-3-coding-text-editor-for-mac-os-x-computers/
- http://www.makeuseof.com/tag/leafpad-ultralightweight-text-editor-linux/
- http://www.makeuseof.com/tag/geany-great-lightweight-code-editor-linux/

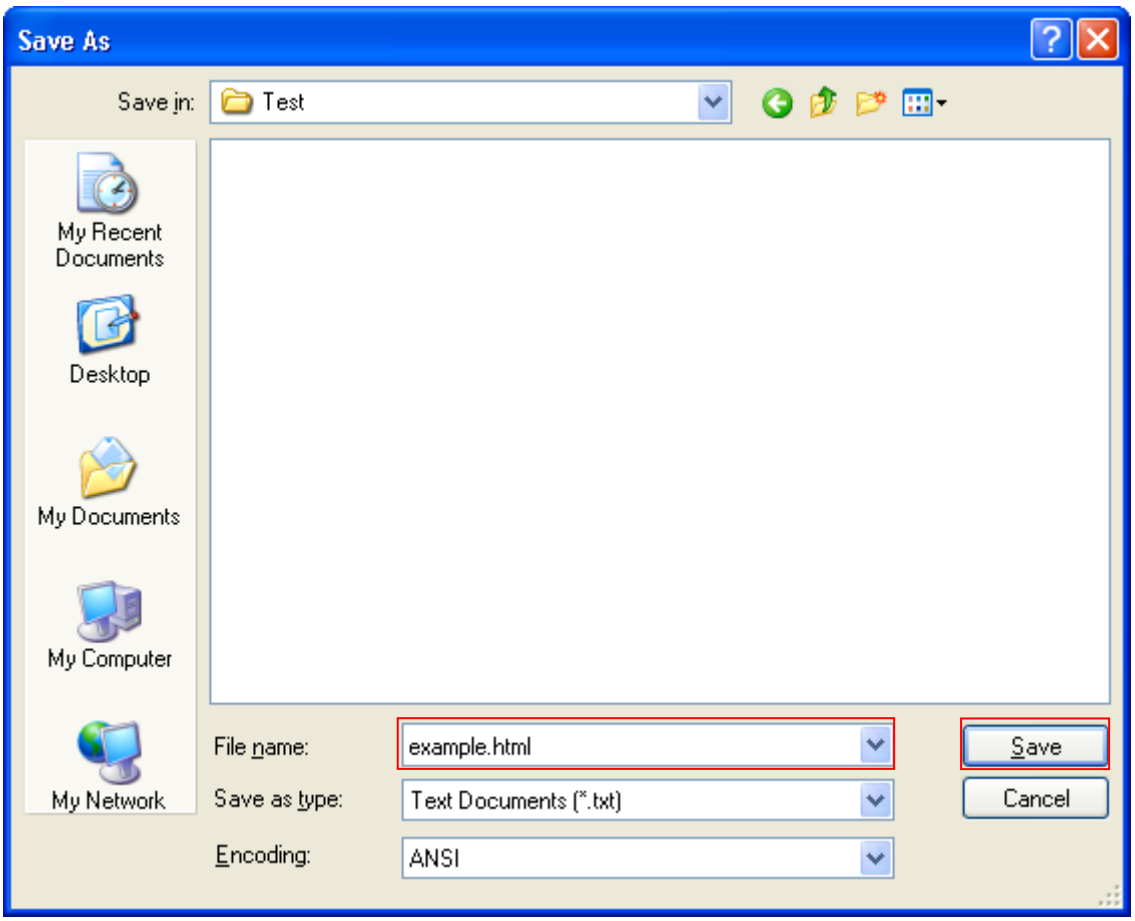
If you'd rather not download any dedicated tools you can still use a text editor like **Notepad** or **Wordpad**. However, the above programs are far better tools for testing and designing, as well as helping you with your coding as it prompts you if you make a mistake or if you are trying to remember the correct word to use. Simple is better, right? I personally use Notepad++ and Microsoft Visual Studio, though I have heard many great things about Expression Studio 4. You'll have to decide what you like best, but all of them work just fine.

NOTE: To test a website created from Notepad or Wordpad:

With the file open, click **File >> Save As**



At the end of the file name type in **.html** and click **Save**



Open the newly saved file (it will, open in your default Internet Browser)

2.1 – Getting to know "the world"

Alright, here's the beginning of the journey. Let's start with just putting something on the screen on this web page. First you'll need to know what **<tags>** are.
XHTML code uses start and end tags to sort out what is going on with each element of the page.

Here is an example of a start tag: **<html>**

Here is an example of an end tag: **</html>**

See the difference? One has the element name enclosed in the pointy brackets and the other is the same but has a slash before the element name.
IMPORTANT: You must close a tag after you open it at some point in the code. Also tags must be nested, that is, meaning that you cannot do the following: **<p> <body> </p> </body>**; it should be **<body> <p> </p> </body>**. See how the tags fit inside each other? Think of them like boxes: you can't put something solid in a box and a half.
The best way to get to know how to program is by actually doing it, so enough theory. Just for a point of reference, I'm going to label each line of code with a number so that I can explain line by line what is going on.

```
1 <?xml version = "1.0 encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5     <body>
6         <p> Hello World, this is my first page! </p>
7     </body>
8 </html>
```

In line 1 I have stated the html code and in line 5 I have ended it. Inside the <html> tag is the <body>, and inside the <body> there is a paragraph (line 3, <p>). If you open this in a web browser, then you will see the following come across the screen:

If you want to change the title of the page from the browser's point of view (eg. firstpage.html) then you can easily add in the following line of code:

<title> Enter Title here </title>

This will make your webpage look more professional.

2.2 – Starting from the <head> and working down the <body>

In most cases, inside the <html> tag there is a <head> and a <body>.
The <head> is usually used for scripting in CSS (Section 3) and JavaScript (explained in an upcoming manual), whereas the <body> is usually the content of the page.

Some content can be changed using the scripting in the <head>, but the <body> is usually the content that is unchangeable on the page. An example would be a short spiel about the website that you are visiting.

You can make changes to the content's formatting using CSS (Section 3) in the <head>. However you can also make changes to the formatting in the <body>.
A commonly used set of tags that are used in the body are the header fonts. These header fonts range in size and strength/boldness. Just see for yourself below:


```
1  <?xml version = "1.0 encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
   Transitional//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
   transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <body>
6          <h1> Hello </h1>
7          <h2> World, </h2>
8          <h3> this </h3>
9          <h4> is my </h4>
10         <h5> first </h5>
11         <h6> page! </h6>
12     </body>
13 </html>
```

2.3 – Is your picture worth a thousand words? – Images

Up to now, we've only talked about text and what it can do on a website, but there's still more. Want to make your website look even more enticing than just fancy fonts? Try getting some good images to make you site really give the audience something to look at. Be careful of copyright laws though; best to take your own pictures if you intend to put your website up on the Internet.
You might need to use Photoshop or some digital imaging skills to create a great picture or perhaps enhance your own image and make it look even more awesome. Try out these guides for some great tips and insight:

- http://www.makeuseof.com/pages/download-idiot-guide-photoshop
- http://www.makeuseof.com/pages/guide-to-digital-photography

The most popular image formats are the following:

- GIF = Graphics Interchange Format
- JPEG = Joint Photographic Experts Group
- PNG = Portable Network Graphics

Take a look at the code below and I'll explain next what it means; that is, how to add images into your webpage.

```
1  <?xml version = "1.0 encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6          <title> How do I add images? </title>
7      </head>
8      <body>
9          <p>
10             <img src = "Picture1.jpg" width = "400" height = "450" alt = "First Picture"
   />
11             <img src = "Picture2.jpg" width = "450" height = "400" alt = "Next Picture">
12             </img>
13         </p>
14     </body>
15 </html>
```

As was taught in previous sections we always start with the <html> and co tags. Next the <head> tag is opened in line 5. Let's just skip down to the important stuff… After the paragraph being opened in line 9 this is where the images are inserted onto the website. To add a picture/image you should use **<img>** to start with. Next you need to suggest where the file is. Usually you would try and have this file in the same folder as the website files else you will have to enter the folder path that it exists in. In the case above I have used **<img src = "Picture.jpg">**. This means that the source (**src**) of the picture resides in the same folder and the name of that picture file is **Picture.jpg**. Easy right?

You don't have to add anything more than **<img src = "Picture.jpg" alt = "something">** to create an image with an **alt** property but you can add properties to it to make some changes to it.

Also known as **alt text**, this property value is displayed when you hover the mouse over the picture.

You may notice that in line 10 I have started the tag with **<img** and ended it with **/>**. This is another way of opening and closing tags. This is the usual way to create images because you can choose the different properties of the image such as width and height as shown in the example above.

In lines 11 and 12 another image is inserted but this is using the other method for opening and closing tags. Line 10 creates the image in a much neater fashion; use that rather than the method in lines 11 and 12.

2.4 – Hyperlinks where can they go?

2.4.1 – Moving around "the world"

Want to show your friends some cool sites on your website but don't know how? You've come to the right place, read on…
Have a look at the code below and see if you can guess what I am doing before I explain it.

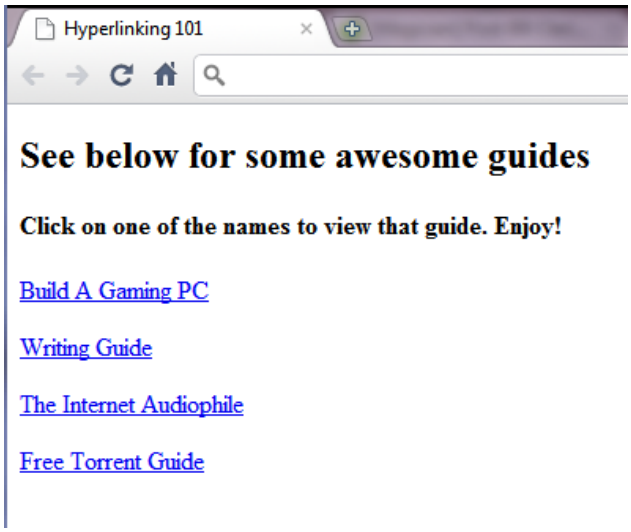
```
1  <?xml version = "1.0 encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6          <title> Hyperlinking 101 </title>
7      </head>
8      <body>
9          <h2> See below for some awesome guides from MakeUseOf.com </h2>
10         <h4> Click on one of the names to view that guide. Enjoy! </h4>
11         <p><a href = "http://www.makeuseof.com/pages/build-a-gaming-pc/">Build A Gaming PC</a></p>
12         <p><a href = "http://www.makeuseof.com/pages/writing-professional-reports-documents/">Writing
13         Guide</a></p>
14         <p><a href = "http://www.makeuseof.com/pages/the-internet-guide-audiophile/">The Internet
15         Audiophile</a></p>
16         <p><a href = "http://www.makeuseof.com/pages/free-torrent-guide/">Free Torrent Guide</a></p>
17     </body>
```

```
16  </html>
```

That's right, I'm creating hyperlinks to some great and useful sites. Basically to hyperlink to a certain web page that has a web address you simply use the below syntax:

** [what you want to hyperlink] **

Doesn't seem very difficult does it? You could quite easily just put text in there like the example code above. However, there is no reason why you couldn't use something else like an image. Just for a bit of extra information: a URL is a Uniform Resource Locator, basically the web address.



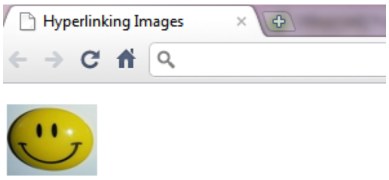
2.4.2 – Pictures remind you of where you've been and take you there again

Here's an example of using an image as a hyperlink:

```
1   <?xml version = "1.0 encoding = "utf-8"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4   <html xmlns="http://www.w3.org/1999/xhtml">
5       <head>
6           <title> Hyperlinking Images </title>
7       </head>
8       <body>
9           <p><a href = "http://www.makeuseof.com">
10              <img src = "button.jpg" width = "70" height = "55" alt = "a button" />
11          </a>
12      </body>
```

```
</html>
```

I'm sure if you're read the previous parts of this section that you release that this is simply mixing creating images and hyperlinks. The syntax is set to have the hyperlinking on the outside and the image on the inside, whereby placing a hyperlink of the inserted image.



2.4.3 – You've got mail – Hyperlinking to an email address

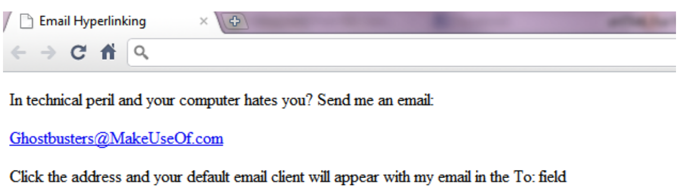
This is simply a repeat of the last part, but if you haven't been paying attention that much then just have a look at the code below:

```
1  <?xml version = "1.0 encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6          <title> Email Hyperlinking </title>
7      </head>
8      <body>
9          <p>In technical peril and your computer hates you? Send me an email:</p>
10         <a href = "mailto:ghostbusters@makeuseof.com">Ghostbusters@MakeUseOf.com</a>
11         <p>
12             Click the address and your default email client will appear with my email in the To: field
13         </p>
14     </body>
15 </html>
```

Instead of using a URL (eg. http://www.something.com) here I'm using an email address which involves putting the following syntax after the equals sign:

"mailto:[youremailaddress]"

Line 10 is the basic example of this concept. So who are you going to email? Ghostbusters!



2.4.4 – Getting around your world – Internal Hyperlinking

Now you can see how you would get around your own website. This is done simply by using your file name as the URL. Therefore you can have a set up of websites like is shown in the diagram below. The syntax you would be using would go something like this:

** Next Page **

2.5 – Are you special? These characters are...

When you type in information that is going to appear on the website like content, then you might need to put in something like a symbol such as the copyright symbol: © or perhaps a less than or greater than symbol. But since the normal symbols are used by the coding syntax, then there had to be another way to get past this little obstacle, and the solution was using an ampersand (&) and then a short code afterwards to tell the computer what symbol to put in. Below is a table with a few examples of special characters from coding:

Character	xHTML code
<	<
>	>
TM	&#8482
©	&copy

For example you might say:

** There are < six rows in the above table, but > 2 rows **

There are < 6 rows in the above table, but > 2 rows

If you want a larger list of the special characters, then please visit the following site:

www.w3.org/TR/REC-html40/shtml/entities.html

2.6 – Lists, lists and more lists

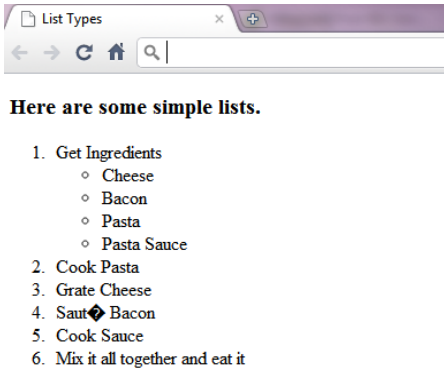
Alright now we're going to have to organise a few things, like a shopping list. There are two types of lists. They are the:

- Ordered List (numbers, alphabets, roman numerals)
- Unordered List (bullet points)

For an ordered list you would use the following tags = ** **
For an unordered list you would use the following tags = ** **
For example:

```
<?xml version = "1.0 encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title> List Types </title>
    </head>
    <body>
        <h3>Here are some simple lists.</h3>
        <ol>
            <li>Get Ingredients</li>
            <ul>
                <li>Cheese</li>
                <li>Bacon</li>
                <li>Pasta</li>
                <li>Pasta Sauce</li>
            </ul>
            <li>Cook Pasta</li>
            <li>Grate Cheese</li>
            <li> Cook Bacon</li>
            <li>Cook Sauce</li>
            <li>Mix it all together and eat it</li>
        </ol>
    </body>
</html>
```

In the above example I included both unordered and ordered list types. But did you notice what else I did? I also included a technique called **Nested Lists**. These nested lists may be used to represent hierarchical relationships, like the list of ingredients in the *Get Ingredients* step of the recipe above.

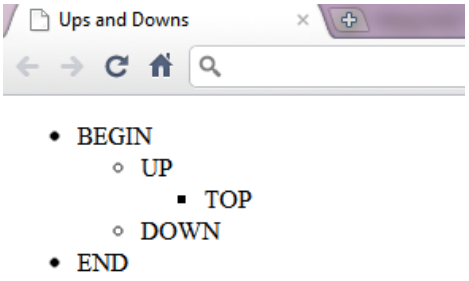


You may see that I started the entire list as an ordered list in Line 10 and finished it in Line 23. In between you would see the **
** and **
** tags I have used. These denote **List Items**. The list items are the words that show up such as in Line 21: **
Cook Sauce
. The words **Cook Sauce would show up next to the number 5 as it is the fifth list item in an ordered list.

If you want to go to the next hierarchical level of dot points or numbers then nest inside themselves like this:

```
1  <?xml version = "1.0 encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6          <title> Ups and Downs </title>
7      </head>
8      <body>
9          <ul>
10             <li>BEGIN</li>
11             <ul>
12                 <li>UP</li>
13                <ul>
14                     <li>TOP</li>
15                 </ul>
16                 <li>DOWN</li>
17             </ul>
18             <li>END</li>
```

```
19          </ul>
20      </body>
21  </html>
```



2.7 – Tables... no not maths

Is this as difficult as your multiplication tables? Of course not, if you go about it the right way. If you're just starting out with this concept and I assume you are, then it is usually best to draw the table you want to make on a piece of paper like I have below:

Product Prices

Product	Price
4pk Energy Drinks	$8.00
1kg Bacon	$10.50
Loaf of Bread	$2.50
Total	$22.00

Now look at it in code below:

```
<?xml version = "1.0 encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title> Tabling </title>
    </head>
    <body>
        <table border>
            <caption><strong> Product Prices </strong></caption>
            <thead>
                <tr><th>Product</th><th>Price</th></tr>
            </thead>
            <tbody>
                <tr><td>4pk Energy Drinks</td><td>$8.00</td></tr>
                <tr><td>1kg Bacon</td><td>$10.50</td></tr>
                <tr><td>Loaf of Bread</td><td>$2.50</td></tr>
            </tbody>
            <tfoot>
                <tr><td>Total</td><td>$22.00</td></tr>
            </tfoot>
        </table>
    </body>
</html>
```

Now mix them together and the display below should help you understand how the table is structured:

<table border>

<caption>Product Prices</caption>

<thead> <tr>	**<td>Product</td>**	**<td>Price</td>**	**</tr> </thead>**
<tbody> <tr>	<td>4pk Energy Drinks</td>	<td>$8.00</td>	</tr>
<tr>	<td>1kg Bacon</td>	<td>$10.50</td>	</tr>
<tr>	<td>Loaf of Bread</td>	<td>$2.50</td>	</tr>
<tfoot> <tr>	**<td>Total</td>**	**<td>$22.00</td>**	**</tr> </tfoot>**

</table>

A <table> or many tables can also be useful as frames in web pages, one for the menu, one for the content and one for the header.

<thead> and <tfoot> bold the first and last row respectively to draw more attention to those parts of the table. Most people would look at the total in the footer of the table first right?

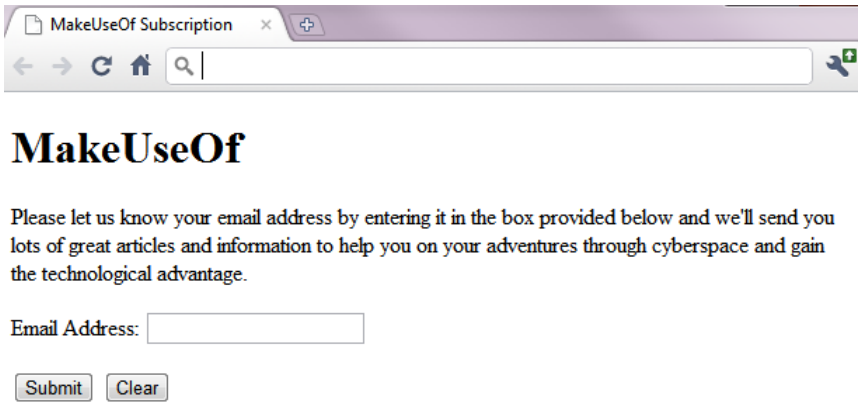
<td> is the table data in the rows of the table.

<tr> is the table rows that start and end on each line of code for neatness. <caption> is good to make sure that your table is not just a bundle of information without a reason for it existing.

2.8 – Digital Forms (Pens away)

When surfing the net you will need to interact with the web pages that you encounter. For example, at www.makeuseof.com you would need to enter your email address as circled below to subscribe to the newsletter and daily updates from MakeUseOf. After you enter your email address you would press **Join** and this would send the information (your email) in the text box next to the button to either a database or perhaps another email address. **Forms** are used to do this which is what you will learn in this chapter.

Below is a form that is used to put just your name in and click either **Submit** or **Clear**:



Here's the code from behind the scenes, which I will explain this in more detail shortly:

```
<?xml version = "1.0 encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>MakeUseOf Subscription</title>
    </head>
    <body>
        <h1>MakeUseOf</h1>
        <p>Please let us know your email address by entering it in the box provided below
and we'll send you lots of
        great  articles and information to help you on your adventures through cyberspace
and gain the technological
        advantage.</p>
        <form method = "post" action = "">
            <p>
                <input type = "hidden" name = "recipient" value =
"subscribe@makeuseof.com" />
                <input type = "hidden" name = "subject" value = "Subscribe Email" />
                <input type = "hidden" name = "redirect" value = "index.html" />
            </p>
            <p><label>Email Address:
                <input name = "name" type = "text" size = "20" maxlength = "60" />
            </label></p>
            <p>
                <input type = "submit" value = "Submit" />
                <input type = "reset" value = "Clear" />
            </p>
        </form>
    </body>
```

```
28  </html>
```

Firstly the most important thing in the above script is Line 10. This is the start of the form. The method is usually either **post** or **get**. Quite self explanatory, but **post** is sending the information somewhere to make a record, such as an email address or database. For example: posting a question on MakeUseOf Answers. **Get**, on the other hand, sends the information you have provided and returns with feedback information, such as a Search Engine, sending the search keywords and returning with the results.

The above coding block is an example of a post form whereby you would enter your email address and it would be sent to the hidden property with an email address after clicking the Submit button. The <label> tag in lines 19-21 creates the text box on the page and gives it the max characters available to be used in that field. Lines 22 – 25 place the Submit and Reset/Clear buttons on the page under the text box. The **Reset** button simply deletes any text entered in the text box or boxes in that form. The **Submit** button follows instructions from the hidden parts of the form which are created in lines 14 – 18. The hidden type would usually assume for something automatic or a part of something else being used in the current form. In this case it is the latter giving the **post**ed information a destination, in this case subscribe@makeuseof.com, with the subject set, in this case "Subscribe Email", and then redirects you to another page, in this case the main page or "index.html".

2.9 – *meta* what? Why?

Ever wondered how Search Engines find websites? Well basically this is what they use: **meta elements**. Search engines usually catalogue sites by following links to pages on sites they find. These meta elements have information about the page on them. Have a look at the following extract from some code for an example:

```
1  <head>
2      <meta name = "keywords" content = "learn, makeuseof how to, computer help,
   technology, muo, tech" />
3      <meta name = "description" content = "lots of great articles and information to help
   you on your adventures
4      through cyberspace and gain the technological advantage." />
5  </head>
```

As you can see above, the meta information goes in the <head> tag and has the types: **keywords** and **description**. The content is the other part of the meta information which is either the keywords or the description as shown in the example.

3 – Design with CSS

Most of the time people who look at guides such as these just happen to like playing video games. CSS however is not Counter Strike Source, nor is it a First Person Shooter (FPS) at all. CSS is a technology that works with xHTML, and stands for **C**ascading **S**tyle **S**heets. xHTML is pretty boring on its own, but if you add a fair serving of CSS the formatting and presentation of your creation is far more interesting. Authors can make changes to elements on a web page such as fonts, spacing, colours; this is done separately from the document structure (head, body, etc.; this will be explained in later chapters). xHTML was actually designed to specify the content and structure of a document . It's not like xHTML couldn't make changes to the formatting of the content. However, this set up is far more beneficial as it can be controlled from one place if required. For example, if a website's format is determined entirely by an attached style sheet, a web designer can simply put in place another style sheet to heavily change the presentation of the website.

3.1 – Inline ~~Dancing~~ Styles

As mentioned above, this section is all about formatting and styles. Since there are many ways to change the style of your content and page, I thought it would be good to start with the most straight forward technique which is **Inline Styles**. This is performed by placing the code in the property section of a tab that encompasses the content. Like this:

```
<element style = "property: value; property2: value2 …..">  CONTENT </element>
```

Sound too difficult? Let me give you an example:

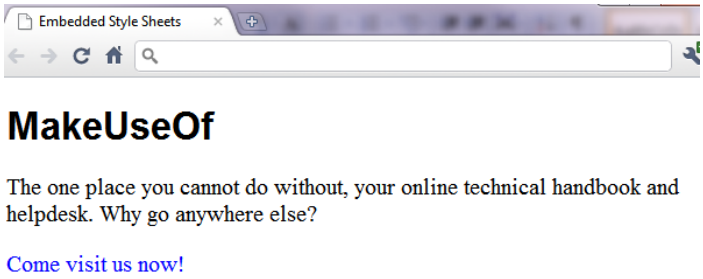
```
<p style = "font-size: 20pt; color: #0000ff"> This is much easier than you think it is. Might
be worth making a list of the different format properties you can change or memorize them if
you can. Just a hint if you didn't guess, this content is size 20pt font and has blue text.
</p>
```

Note: Colour is spelt **color** when using this code since it was created somewhere not as cool as Australia or Canada; I hope it doesn't bother you too much
The bolded information in the example above is the formatting that is being processed on the content encompassed in the <p> tag. For a list of hexadecimal codes for different colours simply search Google or use this site: http://html-color-codes.com/

3.2 – Embedded Style Sheets (Cheat Sheets are win)

Using the inline styles in the previous section can be a pain if you have a very large site. But if you want to use the same styles over and over again then why not use an **Embedded Style Sheet**? This alternative allows you to create your own styles in the <head> tag of the code and then you refer to them in the code when inserting some content on your page. Too complicated? Here's an example:

```
1  <?xml version = "1.0 encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6          <title>Embedded Style Sheets</title>
7          <style type = "text/css">
8              p         { font-size:15pt; color #ff1111 }
9              .xtra   { color: #0000ff }
10             h1       { font-size:25pt; font-family:arial, sans-serif }
11         </style>
12     </head>
13     <body>
14         <h1>MakeUseOf</h1>
15         <p>The one place you cannot do without, your online technical handbook and
   helpdesk. Where else? </p>
16         <p class = "xtra">Come visit us now!</p>
17     </body>
18 </html>
```



See how the text changes colour, size or format depending on the style sheet at the top? This isn't very hard to understand right?

In line 7 where we introduce the start of the <styles> tag with type: **text/css** this is called a MIME (Multipurpose Internet Mail Extensions) type that describes the content existing in that tag/file. CSS documents always use the MIME text **text/css**. Javascript, which will be covered in another volume of this manual, uses the **text/javascript** MIME type. There are many different MIME types, however the main ones are Javascript and CSS.

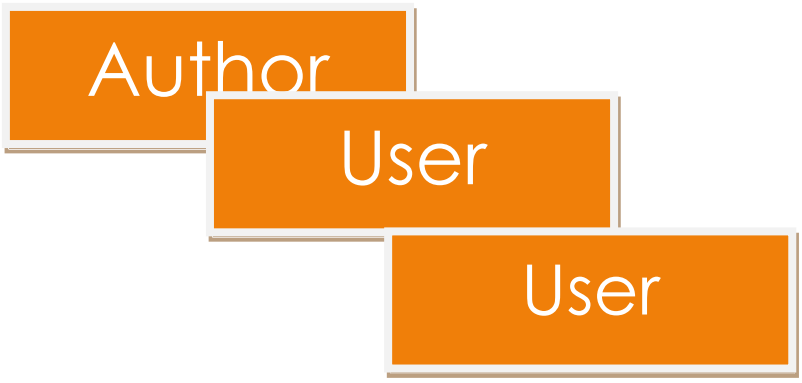
Line 16 uses the **.xtra** class that was made earlier. The way this works is that it adds on the **xtra** class to whatever style it is opened on, whereby overwriting any properties that the **xtra** class uses. For example: if a style has size 20pt font and is the colour green, and a class is put over on it that has a different size of font, then the new font size will replace the old one, but the old colour green will continue as is.

3.3 – Styles at war (conflicting styles)

There are three levels of styles and they are:

- User (viewing the website)
- Author (of the website)
- User Agent (browser)

The styles merge together in such a way that creates the best possible setup from the position of the user. The following chart shows the hierarchy of the three levels:



3.4 – Style Sheets from beyond (external)

Don't you think it would be annoying to have to always write out the same style sheet in each new coding file? There is a solution: **External Style Sheets**. You can create another file with the purpose of using it for formatting; it's a "**.css**" file. To use it in another file simply type in the following extract:

```
1  <head>
2      <link rel = "stylesheet" type = "text/css" href = "filename.css">
3  </head>
```

Replace **filename** with the name of your CSS file and there we go, they are linked. Make sure that your CSS file is in the same folder as your linked file(s).
Sample CSS file:

```
1  /* External Stylesheets */
2  body         { font-family: arial, Helvetica, sans-serif }
3  a.nodec    { text-decoration: none }
4  a.hover     {text-decoration: underline }
5  li em          { font-weight: bold }
6  h1, em      { text-decoration: underline }
7  ul               {margin-left: 20px
```

```
8   ulu l          {font-size: .8em; }
```

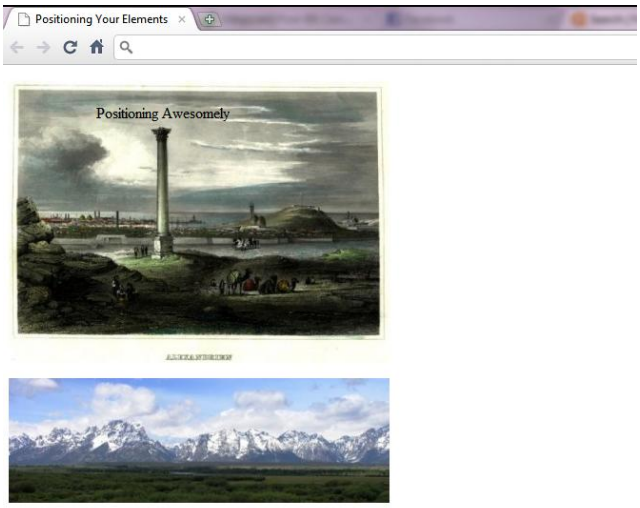
Before we continue I have neglected to mention what em does. Above you'll see in the last line that I have put "ul ul { font-size: .8em; }" and this means that the font-size will be changed to the relative .8 or 80% of the normal size that the user wants it to be using their own style sheet loaded into their browser. Most people do not use a user-defined style sheet so let's not worry about this.

3.5 – Positioning Elements (where to next?)

When you put an image on a web page you don't really want it to just go anywhere. Wouldn't you want to have a say in it? Well this is how you do it, well it's actually an example and I'll explain it shortly:

```
1   <?xml version = "1.0 encoding = "utf-8"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4   <html xmlns="http://www.w3.org/1999/xhtml">
5       <head>
6           <title>Positioning Your Elements</title>
7           <style type = "text/css">
8               .bgpic     {position: absolute;
9                                   top: 0px;
10                                  left: 0px;
11                                  z-index: 1 }
12              .fgpic      { position: absolute;
13                                   top: 25px;
14                                  left: 100px;
15                                  z-index:2 }
16              .txt          { position: absolute;
17                                  top: 25px;
18                                  left: 100px;
19                                  z-index:3
20                                  font-size: 25pt }
21          </style>
22      </head>
23      <body>
24          <p><img src = "bgpic.jpg" class = bgpic" alt = "Image 1" /></p>
25          <p><img src = "fgpic.jpg" class "fgpic" alt = "image 2" /></p>
26          <p class = "txt">Positioning Awesomely</p>
27      </body>
28  </html>
```

In lines 9 to 13 you will notice that it is a class with the ID as **fgpic** and has a few properties used in it. The **position** property is set to **absolute** which means that no matter how the user changes it, the picture will stay where your (the author) places it with their code. The **top** and **left** properties designate a point at which the element (eg. picture/text) will be placed. The **z-index** property is a very powerful tool because it sets the level of stacking as shown in the screenshot below:



See how the background image is at the back with z-index value of 1 and the text is at the front with z-index value of 3, while the foreground image is in the middle with a z-index value of 2. Makes it look quite good actually if you play your cards right

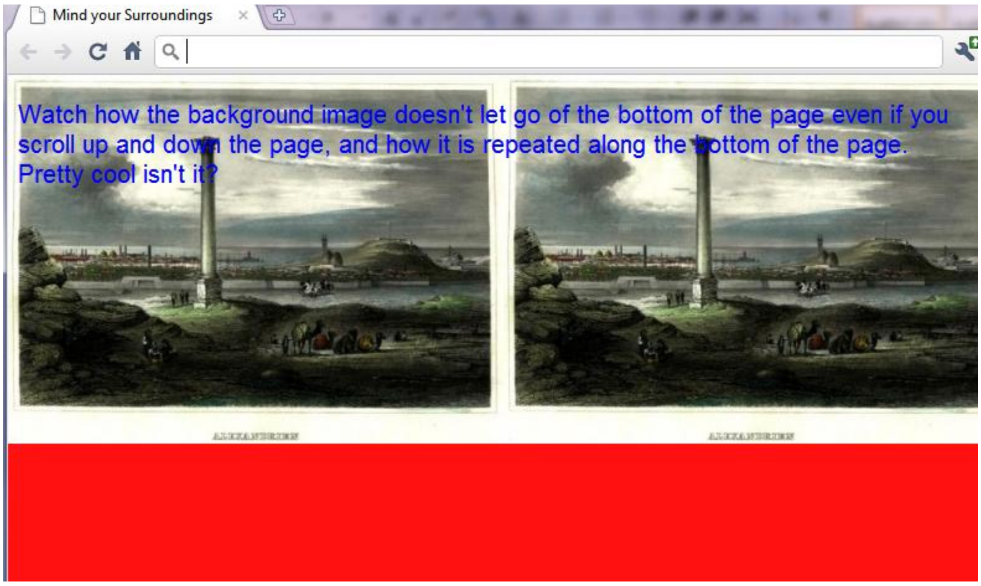
3.6 – Mind your surroundings (background)

Websites look good with backgrounds don't they? Wouldn't it be really boring if all websites just had a white or black background? Why not put a picture in there and change the colour a bit? There are a few properties that you can use to make the background of your page stand out a bit more and give the page some flare. Have a look at the following code and see if you can work out what the highlighted properties do:

```
1  <?xml version = "1.0 encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6          <title>Mind your Surroundings</title>
7          <style type = "text/css">
8              body     { background-image: url(bgpic.jpg);
9                                background-position: bottom left;
10                               background-repeat: repeat-x;
```

```
                              background-attachment:fixed;
                              background-color: #ff1111 }
                p            {font-size: 15pt;
                             color: #0000ff;
                             font-family: arial, sans-serif }
            </style>
        </head>
        <body>
            <p>Watch how the background image doesn't let go of the bottom of the page even
    if you scroll up and down
            the page, and how it is repeated along the bottom of the page. Pretty cool isn't
    it? </p>
        </body>
    </html>
```

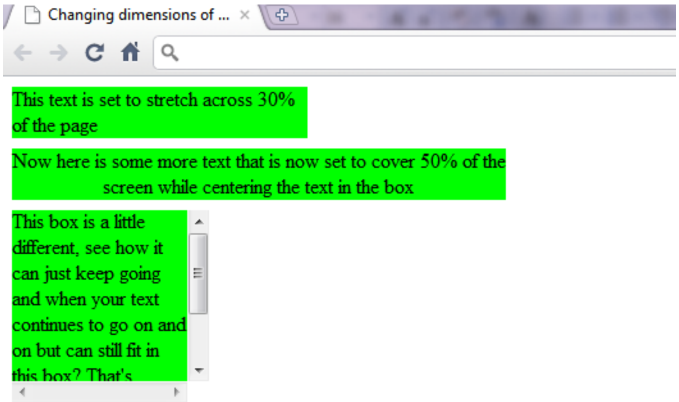
Did you work out what it does? Basically the background image is what we're going to use in the background, the path of the image goes in the brackets/parenthesis like this → url(**HERE**). You might think of this as having z-index value 0 since it is always at the very back of the page. The background position of the image has been set to the bottom left, pretty self explanatory right? Next, the background image has been repeated across the x-axis of the page (repeat-x) and not only that but it is fixed to the bottom of the window (background-attachment). Finally the colour has been randomly set to mainly be red. Have a look below for the result:



3.7 – How big do you think? (dimensions of elements/text limits)

If you think that is all CSS has to offer, you're sorely mistaken. CSS rules can specify the actual dimensions of each page element. Let's take the example of a text box. Do you want to type in some text that does not go all the way across the screen, or

perhaps make a text box that can be scrolled without moving the page? This is where you should be then. See the screenshot below for what I have just described:



Now let's have a look at the code behind the scenes:

```
1  <?xml version = "1.0 encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6          <title>Changing dimensions of text elements</title>
7          <style type = "text/css">
8              div { background-color: #00ff00;
9                       margin-bottom: .5em }
10         </style>
11     </head>
12     <body>
13         <div style = "width: 30%"> This text is set to stretch across 30% of the
   page</div>
14         <div style = "width: 50%; text-align: center">Now here is some more text that is
   now set to cover 50% of the
15         screen while centering the text in the box</div>
16         <div style = "width: 20%; height: 150px; overflow: scroll">This box is a little
   different, see how it can just keep
17         going and when your text continues to go on and on but can still fit in this box?
   That's because we've set the
18         overflow property to scroll!</dir>
19     </body>
20 </html>
```

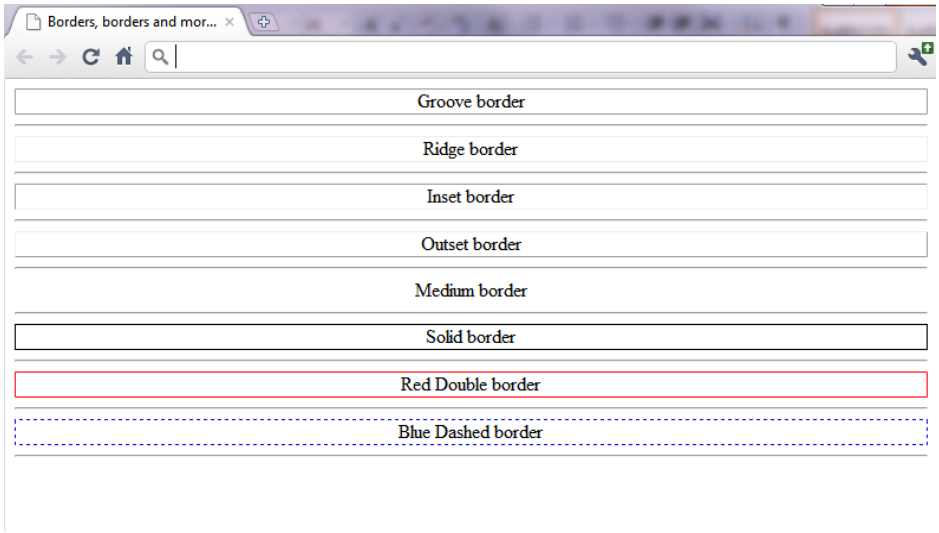
Just a minor note: line 6 adds a marginal border on the bottom of each of the text boxes. Pretty cool, right? But more on borders in the next section.

3.8 – What goes around comes around (borders)

I don't think this needs an explanation but I'll give one anyway. Basically you can put borders around pretty much anything so let's have a look at how to do it. So here's the code:

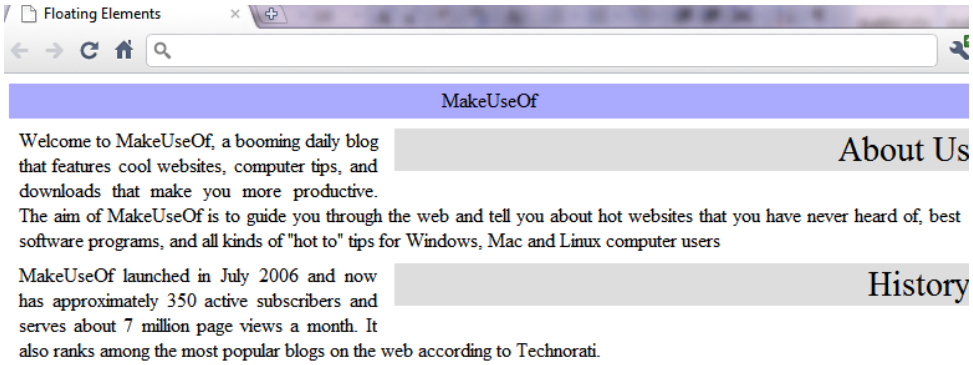
```
<?xml version = "1.0 encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title> Borders, borders and more borders</title>
        <style type = "text/css">
            div         { text-align: center;
                          width: 40%;
                          position: relative;
                          border-width: 1px }
            .groove {border-style: groove }
            .ridge { border-style: ridge }
            .inset { border-style: inset }
            .outset { border-style: outset }
            medium { border-style: medium }
            .thin { border-style: thin }
            .solid {border-style: solid }
            .double { border-style: double }
            .dashed { border-style: dashed }
            .red {border-color: red }
            .blue { border-color: blue }
        </style>
    </head>
    <body>
        <div class = "groove">Groove border</div><hr />
        <div class = "ridge">Ridge border</div><hr />
        <div class = "inset">Inset border</div><hr />
        <div class = "outset">Outset border</div><hr />
        <div class = "medium">Medium border</div><hr />
        <div class = "solid">Solid border</div><hr />
        <div class = "red double">Red Double border</div><hr />
        <div class = "blue dashed">Blue Dashed border</div><hr />
    </body>
</html>
```

Here is what the code does, basically an assortment of borders surrounding the name/s of the type of border being used. Bear in mind that the opposite of groove is ridge and the opposite of inset is outset.



3.9 – Floating and Flowing Elements

It's usually quite boring just to see heading, then text, then heading then text. While not make it look a bit nicer? There's a method that can be used called **floating**, and now I'm going to show you how to do just that. Floating allows you to move an element to one side of the screen while other content in the document then flows around the floated element. The floated element could be a picture or a heading or even another block of text. Now let's have a look at what that looks like:



Pretty good for quite a few situations, now this is the code that constructs this design:

```
1  <?xml version = "1.0 encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
```

```
        <title>Floating Elements</title>
        <style type = "text/css">
            div.heading { background-color: #aaaaff;
                                        text-align: center;
                                        padding: .3em }
            div.floated { background-color: #dddddd;
                                       font-size: 1.75em;
                                       padding: .3m;
                                       margin-left: .5em;
                                       margin-bottom: .5em;
                                       float: right;
                                       text-align: right;
                                       width: 60% }
            p                  { text-align: justify;
                                       margin: .5em }
            div.borders { border: 1px solid #aaaaff }
        </style>
    </head>
    <body>
        <div class = "heading">MakeUseOf</div>
        <div class = "section">
            <div class = "floated">About Us</div>
            <p>Welcome to MakeUseOf, a booming daily blog featuring cool websites,
computer tips, and downloads to
             make you more productive. The aim of MakeUseOf is to guide you through the
web and tell you about hot
             websites that you have never heard of, best software programs, and all kinds
of "hot to" tips for Windows,
             Mac and Linux computer users</p>
        </div>
        <div class = "section">
            <div class = "floated">History</div>
            <p>MakeUseOf launched in July 2006 and now has approximately 350 active
subscribers and serves about 7
            million page views a month. It also ranks among the most popular blogs
according to Technorati.</p>
        </div>
    </body>
</html>
```

Isn't it amazing what you can do if you just find the right method?

3.10 – Don't drop down the menu – example

If you're thinking of creating a website, you're going to most likely need a menu, right? Well this might be the right place for you to go if you want something that isn't just sitting there. Dynamic elements make web pages look better and give a better feel to the overall site.

One of my favourite types of menus has to be a **drop down menu** so now let's have a look at how to make one by using CSS. Check out the code below:

```
<?xml version = "1.0 encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Awesome Drop Down Menu</title>
        <style type = "text/css">
            body                        { font-family: "times new roman", times, serif }
            div.menu                { background-color: #119944;
                                                width: 12em;
                                                border: 3px solid #22aa55;
                                                color: black;
                                                text-align: center;
                                                font-weight: bold }
            div.menu:hover a { display: block }
            div.menu a              { display: none;
                                                 color: #22cc22;
                                                 text-decoration: none;
                                                 border-top: 2px solid #22aa55;
                                                 background-color: white;
                                                 width: 12em }
            div.menu a:hover { background-color: #11aa11 }
        </style>
    </head>
    <body>
        <div class = "menu">Menu
            <a href = "http://www.makeuseof.com">Home</a>
            <a href = "http://www.makeuseof.com/about">About Us</a>
            <a href = "http://www.makeuseof.com/tech-fun">Geeky Fun</a>
            <a href = "http://www.makeuseof.com/pages/">Guides</a>
        </div>
    </body>
```

```
33  </html>
```

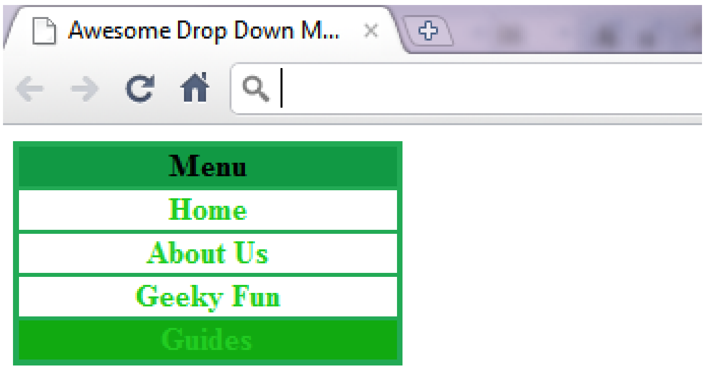
I know this seems a little bit daunting at first, but if you be patient and just read on, you'll understand soon enough.

Line 15 says: when I have a <**div**> tag with class = "**menu**" and the mouse is **hover**ing over it **display** the **block**s inside it.

Lines 16-21 say: when I have a <**div**> tag with class = "**menu**" and an <**a**> tag then set these formats. Bear in mind that these lines choose the format for the hidden menu buttons. Lines 9-14 set up the formats for the menu button to scroll over to show the rest of the menu.

Line 22 says: when I have a <**div**> tag with class = "**menu**" and an <**a**> tag and I **hover** over one of these elements then set the **background-color** to a different green.

Have a look below for the final product:



3.11 – User Style Sheets (you are the centre of the universe)

Users can define their own **user style sheets** to make pages look like they want them to. Just to distinguish between **User Style Sheets** and **Author Style Sheets**. User Styles are external style sheets that users can create themselves which are simply done as CSS files without most of the coding. Here I'll show you one:

```
1  body { font-size: 20pt;
2              color: yellow;
3              background-color: #000080 }
```

Wasn't that extremely simple?

If you want to know how to set this up in your own browser you would simply go to **Tools >> Internet Options >> General >> Accessibility >> Then define your own file** The Author Style Sheet is defined inside the code in between <style type = "text/css"> and </style>.

4 – More Information

4.1 – Why use xHTML and co. over design and other applications?

Before you look at this as fact or something similar to that, you should know that this is simply a point of view depending on where you stand and how technically minded you may be. I enjoy using programming languages to complete my projects as it means you can understand what is behind the designs, whereas using design applications like Adobe Dreamweaver and Microsoft FrontPage allow you to create your website using only the tools available on the menus. Therefore, the design applications are limited to the menu option provided to you. In conclusion, it's completely obvious that using programming languages would build the website or completed product into something far more appealing as its functionality is only limited by the programmer's skill with the designated language (eg. JavaScript, CSS, xHTML). I know you're probably thinking that I'm biased, but you will have to just try both out and decide how much effort you want to put into your work then choose your appropriate tools to get to your destination. You might even choose to use both since both Dreamweaver and FrontPage have a "coding view" and a "design view".

There are other ways you can create websites like using Joomla and WordPress.

4.2 – Joomla

Joomla is a great Content Management System (CMS) with a lot of flexibility and with an easy-to-use user interface that a lot of people get intimidated about when they realize how many options and configurations are available. Joomla is a platform based on PHP and MySQL. This software is open-source which you can get from http://www.joomla.org/download.html

If you want in in-depth guide on Joomla try this guide from MakeUseOf:

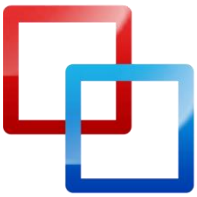
- http://www.makeuseof.com/pages/download-the-complete-beginners-guide-to-joomla

4.3 – WordPress

WordPress is a Content Management System (CMS) that allows users to create and maintain a website through an administrative interface, including an automatically-generated navigation structure, without needing to know HTML or learn any other tool. WordPress is a piece of open source software created by thousands of programmers over the world and put into the public domain, so you do not have to pay to use it. WordPress is a web-based application, written in PHP and MySQL,

designed to run on Linux servers: PHP is a programming language for web applications, MySQL is a relational database (such as MS Access), and Linux is an operating system for web servers – all of these are also open source. WordPress is, by far, the most popular CMS with over 200 million sites worldwide as of late 2009.
For a guide that will let you understand this great too try this website:

- http://www.makeuseof.com/pages/download-build-your-own-wordpress-site



makeuseof.com

Did you like this PDF Guide? Then why not visit MakeUseOf.com for daily posts on cool websites, free software and internet tips.

If you want more great guides like this, why not **subscribe to MakeUseOf and receive instant access to 20+ PDF Guides** like this one covering wide range of topics. Moreover, you will be able to download free Cheat Sheets, Free Giveaways and other cool things.

Subscribe to MakeUseOf : http://www.makeuseof.com/join

MakeUseOf Links:

Home:	http://www.makeuseof.com
MakeUseOf Directory:	http://www.makeuseof.com/dir
MakeUseOf Answers:	http://www.makeuseof.com/answers
Geeky Fun:	http://www.makeuseof.com/tech-fun
PDF Guides:	http://www.makeuseof.com/pages/
Tech Deals:	http://www.makeuseof.com/pages/hot-tech-deals

Follow MakeUseOf:

RSS Feed:	http://feedproxy.google.com/Makeuseof
Newsletter:	http://www.makeuseof.com/join
Facebook:	http://www.facebook.com/makeuseof
Twitter:	http://www.twitter.com/Makeuseof

Download Other MakeUseOf PDF Guides!

Like us to download: http://www.facebook.com/makeuseof