Basic HTML | Adding a touch of style

More advanced features

***Dave Raggett*.**

Having mastered the basics, it is time to move on to more advanced features. The following will teach you how to:

- force line breaks
- introduce non-breaking spaces
- use entities for special characters
- link into the middle of pages
- use preformatted text
- flow text around images
- define clickable regions within images
- create tables
- use roll-overs and other tricks
- enable users to listen to sound files

p.s. I recommend you use HTML Tidy to keep your markup clean and free of errors.

How to force line breaks

Just occasionally, you will want to force a line break. You do this using the ***br*** element, for example when you want to include a postal address:

```
<p>The Willows<br>
21 Runnymede Avenue<br>
Morton-in-the-marsh<br>
Oxfordshire OX27 3BQ</p>
```

The ***br*** element never needs an end-tag. In general, elements that don't take end-tags are known as *empty* elements.

How to introduce non-breaking spaces

Browsers automatically wrap text to fit within the margins. Line breaks can be introduced wherever space characters appear in the markup. Sometimes you will want to prevent the browser from wrapping text between two particular words. For instance between two words in a brand name such as "Coca Cola". The trick is to use *** *** in place of the space character, for example:

```
Sweetened carbonated beverages, such as Coca Cola,
have attained world-wide popularity.
```

It is bad practice to use repeated non-breaking spaces to indent text. Instead, you are advised to set the indent via style rules.

How to use entities for special characters

For copyright notices, or trademarks it is customary to include the appropriate signs:

Symbol	Entity	Example
Copyright sign	©	Copyright © 1999 W3C
Registered trademark	®	MagiCo ®
Trademark	™	Webfarer™

Note HTML 4.0 defines ™ for the trademark sign but this is not yet as widely supported as ™

There are a number of other entities you may find useful:

Symbol	Entity	Example
Less than	<	<
Greater than	>	>
Ampersand	&	&
nonbreaking space		
em dash	—	—
quotation mark	"	"

And then, there are entities for accented characters and miscellaneous symbols in the Latin-1 character set:

			Ð	Ð	Ð
¡	¡	¡	Ñ	Ñ	Ñ
¢	¢	¢	Ò	Ò	Ò
£	£	£	Ó	Ó	Ó
¤	¤	¤	Ô	Ô	Ô
¥	¥	¥	Õ	Õ	Õ
¦	¦	¦	Ö	Ö	Ö
§	§	§	×	×	×
¨	¨	¨	Ø	Ø	Ø
©	©	©	Ù	Ù	Ù
ª	ª	ª	Ú	Ú	Ú
«	«	«	Û	Û	Û
¬	¬	¬	Ü	Ü	Ü
-			Ý	Ý	Ý
®	®	®	Þ	Þ	Þ
¯	¯	¯	ß	ß	ß
°	°	°	à	à	à
±	±	±	á	á	á
²	²	²	â	â	â

³	³	³	ã	ã	ã
´	´	´	ä	ä	ä
µ	µ	µ	å	å	å
¶	¶	¶	æ	æ	æ
·	·	·	ç	ç	ç
¸	¸	¸	è	è	è
¹	¹	¹	é	é	é
º	º	º	ê	ê	ê
»	»	»	ë	ë	ë
¼	¼	¼	ì	ì	ì
½	½	½	í	í	í
¾	¾	¾	î	î	î
¿	¿	¿	ï	ï	ï
À	À	À	ð	ð	ð
Á	Á	Á	ñ	ñ	ñ
Â	Â	Â	ò	ò	ò
Ã	Ã	Ã	ó	ó	ó
Ä	Ä	Ä	ô	ô	ô
Å	Å	Å	õ	õ	õ
Æ	Æ	Æ	ö	ö	ö
Ç	Ç	Ç	÷	÷	÷
È	È	È	ø	ø	ø
É	É	É	ù	ù	ù
Ê	Ê	Ê	ú	ú	ú
Ë	Ë	Ë	û	û	û
Ì	Ì	Ì	ü	ü	ü
Í	Í	Í	ý	ý	ý
Î	Î	Î	þ	þ	þ
Ï	Ï	Ï	ÿ	ÿ	ÿ

You can also use numeric character entities for the greek letters and mathematical symbols defined in Unicode. For more details, take a look at the list specified in the HTML 4 specification. Note that the entity names for these characters aren't recognized in Navigator 4, so you are recommended to stick to the numeric entities instead.

Linking into the middle of Web pages

Imagine you have written a long Web page with a table of contents near the start. How do you make the entries in the table contents into hypertext links to the corresponding sections?

Let's assume that each section starts with a heading, for instance:

```
<h2>Local Night Spots</h2>
```

You make the heading into a potential target for a hypertext link by enclosing its contents with
`<a name=identifier> .... </a>`

```
<h2><a name="night-spots">Local Night Spots</a></h2>
```

The name attribute specifies the name you will use to identify the link target, in this case: "night-spots". The table of contents can now include a hypertext link using this name, for instance:

```
<ul>
  ...
  <li><a href="#night-spots">Local Night Spots</a></li>
  ...
</ul>
```

The # character is needed before the target name. If the target is in a different document, then you need to place the web address of that document before the # character. For example if the document is in "http://www.bath.co.uk/", then the link becomes:

```
<a href="http://www.bath.co.uk/#night-spots">Local Night Spots</a>
```

In the future, you will eventually be able to define link targets without the need for the <*a*> element. The new method is much easier, as all you need to do is to add an *id* attribute to the heading, for instance:

```
<h2 id="night-spots">Local Night Spots</h2>
```

This method doesn't work for 4th generation or earlier browsers, so it should be used with care while these browsers are still in use!

Preformatted Text

One of the advantage of the Web is that text is automatically wrapped into lines fitting within the current window size. Sometimes though, you will want to disable this behavior. For example when including samples of program code. You do this using the *pre* element. For instance:

```
<pre>
    void Node::Remove()
    {
        if (prev)
            prev->next = next;
        else if (parent)
            parent->SetContent(null);

        if (next)
            next->prev = prev;

        parent = null;
    }
</pre>
```

Which renders as:

```
    void Node::Remove()
    {
        if (prev)
            prev->next = next;
        else if (parent)
            parent->SetContent(null);

        if (next)
            next->prev = prev;

        parent = null;
```

```
    }
```

The text and background colors were set via the style sheet. Note that all line breaks and spaces are rendered exactly as they appear in the HTML. The exception is a newline immediately after the start tag *<pre>* and immediately before the end tag *</pre>*, which are discarded. This means that the following two examples are rendered identically:

```
<pre>preformatted text</pre>

<pre>
preformatted text
</pre>
```

Preformatted text is generally rendered using a monospaced font where each character has the same width. If you define a style rule for the *pre* element, some browsers forget to use the monospace font, necessitating the use of the font-family property. For instance if you want to render all pre elements in green you can define the style rule:

```
<style type="text/css">
  pre { color: green; background: white; font-family: monospace; }
</style>
```

When setting the foreground color for text, you are advised to also set the color for the background. This will prevent accidents where the background color is hard to distinguish from the foreground. Rather than setting the background color on the pre element, you may find it more convenient to set it on the body element, for instance:

```
<style type="text/css">
  body { color: black; background: white; }
  pre { color: green; font-family: monospace; }
</style>
```

Flowing text around images

With HTML, you can choose whether any given image is treated as part of the current text line or is floated to the left or right margins. You control this via the *align* attribute. If the align attribute is set to *left* the image floats to the left margin. If it is set to *right* the image floats to the right margin. For instance:

```
<p><img src="sun.jpg" alt="sunburst graphic"
width="32" height="21" align="left"> This text will be
flowed around the right side of the graphic.</p>
```

which renders as:

 This text will be flowed around the right side of the graphic.

The following uses align=*"right"*

```
<p><img src="sun.jpg" alt="sunburst graphic"
width="32" height="21" align="right"> This text will be
flowed around the left side of the graphic.</p>
```

which renders as:

This text will be flowed around the left side of the graphic. 

To force rendering to continue below the floated image you can use the *<br clear=all>* element, for example:

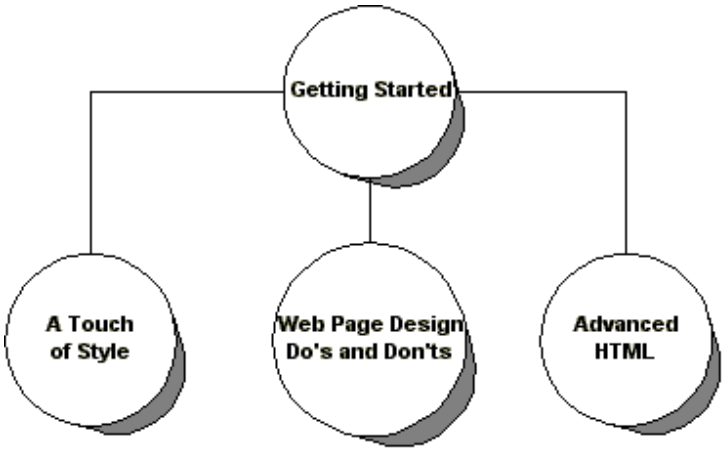
```
<p><img src="sun.jpg" alt="sunburst graphic"
width="32" height="21" align="left"> This text will be
flowed around the right side of the graphic.<br clear="all">
This starts a new line below the floated image.</p>
```

which renders as:

 This text will be flowed around the right side of the graphic.
This starts a new line below the floated image.

Clickable regions within images

The following image acts as a map of a group of Web pages. You can click on the circles to go to the corresponding page.



The markup for this is as follows:

```
<p align="center">
   <img src="pages.gif" width="384" height="245"
      alt="site map" usemap="#sitemap" border="0">
   <map name="sitemap">
      <area shape="circle" coords="186,44,45"
         href="Overview.html" alt="Getting Started">
      <area shape="circle" coords="42,171,45"
         href="Style.html" alt="A Touch of Style">
      <area shape="circle" coords="186,171,45"
         alt="Web Page Design">
      <area shape="circle" coords="318,173,45"
         href="Advanced.html" alt="Advanced HTML">
   </map>
</p>
```

The *src* attribute on the img element specifies the image "pages.gif". The *usemap* attribute references a map element. It uses a Web address to do so, hence the # character. The *border* attribute is set to "0" to suppress the blue border around the image.

The map element specifies which regions in the image act as hypertext links. The *name* attribute matches *usemap* attribute on the img element and acts much like the name attribute on the <a> element. In practice, the map element needs to be in the same file as the img element.

The area element is used to define a region on the image and to bind it to a Web address. The *shape* attribute specifies "rect", "circle" or "poly". The *coords* attribute specifies the coordinates for the region depending on the shape.

- rect: *left-x, top-y, right-x, bottom-y*
- circle: *center-x, center-y, radius*
- poly: $x_1,y_1,\ x_2,y_2,\ \ldots\ x_n,y_n$

The top left pixel is considered as the origin of the image with x and y both equal to zero, x increases rightwards across the image and y increases downwards. Most image manipulation tools allow you to find the pixel coordinates of any given point in the image.

If two or more defined regions overlap, the region-defining element that appears earliest in the document takes precedence (i.e., responds to user input). For a complex shape such as an anular ring, you can make part of a region inactive by overlaying it with another region using the *nohref* attribute, for example:

```
<area shape="circle" coords="186,44,50" nohref>
<area shape="circle" coords="186,44,100"
   href="Overview.html" alt="Getting Started">
```

Where the first circle creates an inactive region within the larger circle created by the second area element. To have any effect, the inactive shape needs to be placed first as otherwise it will be hidden by the active shape.

Why you need to specify the *alt* attribute

The *alt* attribute on the area element is used to supply a text label for the link. Without it the image map is inaccessible to people who can't see the image.

Tables

Tables are used for information as well as for layout. You can stretch tables to fill the margins, specify a fixed width or leave it to the browser to automatically size the table to match the contents.

Tables consist of one or more rows of table cells. Here is a simple example:

Year	Sales
2000	$18M
2001	$25M
2002	$36M

The markup for this is:

```
<table border="1">
<tr><th>Year</th><th>Sales</th></tr>
<tr><td>2000</td><td>$18M</td></tr>
<tr><td>2001</td><td>$25M</td></tr>
<tr><td>2002</td><td>$36M</td></tr>
</table>
```

The *table* element acts as the container for the table. The *border* attribute specifies the border width in pixels. The *tr* element acts as a container for each table row. The *th* and *td* elements act as containers for heading and data cells respectively.

Cell Padding

You can increase the amount of padding for all cells using the *cellpadding* attribute on the table element. For instance, to set the padding to 10 pixels:

```
<table border="1" cellpadding="10">
```

this has the effect:

Year	Sales
2000	$18M
2001	$25M
2002	$36M

Cell Spacing

By contrast the *cellspacing* attribute sets the space between the cells. Setting the cell spacing to 10:

```
<table border="1" cellpadding="10" cellspacing="10">
```

has the effect:

Year	Sales
2000	$18M
2001	$25M
2002	$36M

Table Width

You can set the width of the table using the *width* attribute. The value is either the width in pixels or a percentage value representing the percentage of the space available between the left and right margins. For instance to set the width to 80% of the margins:

```
<table border="1" cellpadding="10" width="80%">
```

which has the effect:

Year	Sales
2000	$18M
2001	$25M

2002	$36M

Text Alignment within Cells

By default browsers center heading cells (th), and left align data cells (td). You can change alignment using the *align* attribute, which can be added to each cell or to the row (tr element). It is used with the values "left", "center" or "right":

```
<table border="1" cellpadding="10" width="80%">
<tr align="center"><th>Year</th><th>Sales</th></tr>
<tr align="center"><td>2000</td><td>$18M</td></tr>
<tr align="center"><td>2001</td><td>$25M</td></tr>
<tr align="center"><td>2002</td><td>$36M</td></tr>
</table>
```

with the following result:

Year	Sales
2000	$18M
2001	$25M
2002	$36M

The *valign* attribute plays a similar role for the vertical alignment of cell content. It is used with the values "top", "middle" or "bottom", and can be added to each cell or row. By default, heading cells (th) position their content in the middle of the cells while data cells align their content at the top of each cell.

Empty Cells

One quirk is the way browsers deal with empty cells, compare:

Year	Sales
2000	$18M
2001	$25M
2002	$36M
2003	

with

Year	Sales

2000	$18M
2001	$25M
2002	

The former occurs when a cell is empty:

```
<td></td>
```

To prevent this, include a non-breaking space:

```
<td> </td>
```

Cells that span more than one row or column

Let's extend the above example to break out sales by north and south sales regions:

Year	Sales		
	North	South	Total
2000	$10M	$8M	$18M
2001	$14M	$11M	$25M

The heading "Year" now spans two rows, while the heading "Sales" spans three columns. This is done by setting the *rowspan* and *colspan* attributes respectively. The markup for the above is:

```
<table border="1" cellpadding="10" width="80%">
<tr align="center"><th rowspan="2">Year</th><th colspan="3">Sales</th></tr>
<tr align="center"><th>North</th><th>South</th><th>Total</th></tr>
<tr align="center"><td>2000</td><td>$10M</td><td>$8M</td><td>$18M</td></tr>
<tr align="center"><td>2001</td><td>$14M</td><td>$11M</td><td>$25M</td></tr>
</table>
```

You can simplify this by taking advantage of the fact that browsers don't need the end tags for table cells and rows:

```
<table border="1" cellpadding="10" width="80%">
<tr align="center"><th rowspan="2">Year<th colspan="3">Sales
<tr align="center"><th>North<th>South<th>Total
<tr align="center"><td>2000<td>$10M<td>$8M<td>$18M
<tr align="center"><td>2001<td>$14M<td>$11M<td>$25M
</table>
```

Notice that as the heading "Year" spans two rows, the first th element on the second row appears on the second rather than the first column.

Borderless tables

These are commonly used for laying out pages in a gridded fashion. All you need to do is to add *border="0"* and *cellspacing="0"* to the table element:

Year	Sales
2000	$18M
2001	$25M
2002	$36M

This was produced using the following markup:

```
<table border="0" cellspacing="0" cellpadding="10">
<tr><th>Year</th><th>Sales</th></tr>
<tr><td>2000</td><td>$18M</td></tr>
<tr><td>2001</td><td>$25M</td></tr>
<tr><td>2002</td><td>$36M</td></tr>
</table>
```

If you leave out the cellspacing attribute you will get a thin gap between the cells, as shown below:

Year	Sales
2000	$18M
2001	$25M
2002	$36M

Coloring your tables

This page uses a style sheet to set the background colors for tables, with a different color for heading and data cells. The style rules I used are as follows:

```
table {
  margin-left: -4%;
  font-family: sans-serif;
  background: white;
  border-width: 2;
  border-color: white;
}
th { font-family: sans-serif; background: rgb(204, 204, 153) }
td { font-family: sans-serif; background: rgb(255, 255, 153) }
```

The last two lines above set the background color for th and td cells to given red/green/blue values. The numbers are in the range 0 to 255 (fully saturated).

Another way to set the background color is to use the *bgcolor* attribute. This works with nearly all browsers, and doesn't rely on style sheet support. The first step is to determine the hexadecimal values for the red, green and blue components of the color you wish to use. A convertor is included in the style page.

```
<table border="0" cellspacing="0" cellpadding="10">
  <tr>
    <th bgcolor="#CCCC99">Year</th>
    <th bgcolor="#CCCC99">Sales</th>
```

```
  </tr>
  <tr>
    <td bgcolor="#FFFF66">2000</td>
    <td bgcolor="#FFFF66">$18M</td>
  </tr>
  <tr>
    <td bgcolor="#FFFF66">2001</td>
    <td bgcolor="#FFFF66">$25M</td>
  </tr>
  <tr>
    <td bgcolor="#FFFF66">2002</td>
    <td bgcolor="#FFFF66">$36M</td>
  </tr>
</table>
```

Centering your tables

You can position your tables midway between the left and right margins by using some CSS. If your style sheet includes the following rule, then all tables will be centered:

```
table {
   margin-left: auto;
   margin-right: auto;
}
```

You can make this specific to a given table by giving it an id value, or by setting a class. The following example applies to tables with a class attribute value of centered:

First here is the style rule:

```
table.centered {
   margin-left: auto;
   margin-right: auto;
}
```

and here is the table markup:

```
<table class="centered" border="1">
<tr><th>Year</th><th>Sales</th></tr>
<tr><td>2000</td><td>$18M</td></tr>
<tr><td>2001</td><td>$25M</td></tr>
<tr><td>2002</td><td>$36M</td></tr>
</table>
```

and here is how it is rendered in your browser:

Year	Sales
2000	$18M
2001	$25M
2002	$36M

Note that you can replace the border attribute by CSS rules for greater control over the appearence of table and cell borders. See the style guide for examples of how to set border styles.

Making your tables accessible

If you are unable to see the table it can be quite hard to understand what the table is about. The first step is to add information describing the purpose and structure of the table. The caption element allows you to provide a caption, and to position this above or below the table. The caption element should appear before the tr element for the first row.

Projected sales revenue by year

Year	Sales
2000	$18M
2001	$25M

which was produced by the following markup:

```
<table border="1" cellpadding="10" width="80%">
<caption>Projected sales revenue by year</caption>
<tr align="center">
  <th>Year</th><th>Sales</th>
</tr>
<tr align="center"><td>2000</td><td>$18M</td></tr>
<tr align="center"><td>2001</td><td>$25M</td></tr>
</table>
```

Here is the same table with `align="bottom"` added to the caption element:

Year	Sales
2000	$18M
2001	$25M

Projected sales revenue by year

The table element's *summary* attribute should be used to describe the structure of the table for people who can't see the table. For instance: "the first column gives the year and the second, the revenue for that year".

```
<table summary="the first column gives the year
and the second, the revenue for that year">
```

Specifying the relation between header and data cells

When a table is rendered to audio or to Braille, it is useful to be able to identify which headers describe each cell. For instance, an aural browser could allow you to move up and down or left and right from cell to cell, with the appropriate headers spoken before each cell.

To support this you need to annotate the header and/or data cells. The simplest approach is to add the *scope* attribute to header cells. It may be used with the following values:

- **row**: The current cell provides header information for the rest of the row that contains it.
- **col**: The current cell provides header information for the rest of the column that contains it.

Applying this to the example table gives:

```
<table border="1" cellpadding="10" width="80%">
<caption>Projected sales revenue by year</caption>
<tr align="center">
  <th scope="col">Year</th>
  <th scope="col">Sales</th>
</tr>
<tr align="center"><td>2000</td><td>$18M</td></tr>
<tr align="center"><td>2001</td><td>$25M</td></tr>
</table>
```

For more complex tables, you can use the *headers* attribute on individual data cells to provide a space separated list of identifiers for header cells. Each such header cell must have an *id* attribute with a matching identifier.

A final point is that you should consider using the *abbr* attribute to specify an abbreviation for long headers. This makes it tolerable to listen to lists of headers for each cell, for instance:

```
<th abbr="W3C">World Wide Web Consortium</th>
```

Roll-Overs and other tricks

A little JavaScript can go a long way to enliven your pages. You will be shown below how to create "rollovers" where the appearence of a link changes as you move the mouse over it. You will also learn how to create cycling banner ads which help to direct visitors to your sponsors' sites

Roll-Overs

In the most common form, a roll-over consists of an image serving as a hypertext link. While the mouse pointer is over the image, it changes appearence to attract attention to the link. For example, you could add a glow effect, a drop shadow or simply change the background color. Here is an example:

```
<script type="text/javascript">
if (document.images)
{
    image1 = new Image;
    image2 = new Image;
    image1.src = "enter1.gif";
    image2.src = "enter2.gif";
}

function chgImg(name, image)
{
    if (document.images)
    {
        document[name].src = eval(image+".src");
    }
}
</script>

...

<a href="/" onMouseOver='chgImg("enter", "image2")'
onMouseOut='chgImg("enter", "image1")'><img name="enter"
src="enter1.gif" border="0" alt="Enter if you dare!"></a>
```

and here is what it looks like ...

I created these images using a freeware painting tool by adding a hot wax effect and then a drop shadow to the text. You can find lots of advice and royalty free clipart on the Web via most search engines.

Banner Ads

If your website has several sponsors, then you can use an image link that cycles through each of the sponsors in turn. The first step is to create an image for each of your sponsors. All the images should have the same size. The corresponding URLs for the images and for the websites are then placed into the arrays named *adImages* and *adURLs* defined at the start of the script. The img element for the link should be initialized to the first image in the array. The cycle is started off using the onload event on the body element.

```
<html>
<head>
<title>cycling banner ads</title>
<script type="text/javascript">
if (document.images)
{
    adImages = new Array("hosts/csail.gif",
               "hosts/ercim.gif", "hosts/keio.gif");
    adURLs = new Array("www.csail.mit.edu",
               "www.ercim.org", "www.keio.ac.jp");
    thisAd = 0;
}

function cycleAds()
{
    if (document.images)
    {
        if (document.adBanner.complete)
        {
            if (++thisAd == adImages.length)
                thisAd = 0;

            document.adBanner.src = adImages[thisAd];
        }
    }

    // change to next sponsor every 3 seconds
    setTimeout("cycleAds()", 3000);
}

function gotoAd()
{
    document.location.href = "http://" + adURLs[thisAd];
}
</script>
</head>
<body onload="cycleAds()">
...

<a href="javascript:gotoAd()"><img name="adBanner"
src="hosts/csail.gif" border="0" alt="Our sponsors"></a>
```

Our Sponsors: 

Note: you are recommended to make sure that all of the images are the same width and height. An alternative is to add width and height attributes to the img element to ensure the images are all shown at the same size.

What about browsers that don't support scripting?

The content of a *noscript* element is only shown if the browser doesn't support scripting. It should be used when you want to give people access to information that would otherwise be inaccessible to people with browsers that don't support scripting. Let's assume that you want to make the links for your sponsors available as text:

```
<noscript>
Our sponsors: <a href="http://www.lcs.mit.edu/">MIT</a>,
<a href="http://www.inria.fr/">INRIA</a>, and
<a href="http://www.keio.ac.jp/">Keio University</a>.
</noscript>
```

There are many free sources of information about scripting, which can be easily found via most search engines.

Enable users to listen to sound files

Let's assume that you and your friends have got together to record some music in your garage, and you now want to get this out to the listening public. The first step is to compress the recorded audio, e.g. as an mp3 file and upload it to your website. For explanatory purposes, let's assume that this is then available at: `http://example.com/music/myband.mp3`. In the examples below you should replace this with the correct location for your website.

The next step is to create a playlist file with the file extension .m3u. This avoids the lengthy wait before users start to hear the music that typically occurs if you link directly to the mp3 file. You can create the playlist with a text editor and it just needs to include the URL for the mp3 file. For the example sound file, this would be:

```
http://example.com/music/myband.mp3
```

Upload the m3u playlist file to your web server. You can now add a link to your band's web page as follows:

```
<a href="http://example.com/music/myband.m3u"
type="audio/x-mpegurl">listen to our band</a>
```

You may also need to check with the web server adminstrator that the correct MIME types are set for the both mp3 and m3u files.

- m3u extension with audio/x-mpegurl
- mp3 extension with audio/mpeg

Note the above approach works best for people with broadband connections. You should consider providing a lower quality version of the mp3 file for people on low speed connections (i.e. at a much lower data rate than 128K).

A very similar approach can also be used for Ogg Vorbis sound files (MIME type "application/ogg" and file extension ".ogg"). These can be used with either m3u or pls playlist files, but not all music players will be configured to support this. Consult a search engine for more details on this and other audio formats.

Getting Further Information

W3C's Recommendation for HTML 4.0 is the authoritative specification for HTML. However, it is a technical specification. For a less technical source of information you may want to purchase one of the many books on HTML, for example "Raggett on HTML 4", published 1998 by Addison Wesley. XHTML 1.0 is now a W3C Recommendation.

Best of luck and get writing!

Dave Raggett <*dsr@w3.org*>