



Advanced HTML | Adding a touch of style



Getting started with HTML

Dave Raggett, revised 13th February 2002.

This is a short introduction to writing HTML. Many people still write HTML by hand using tools such as NotePad on Windows, or SimpleText on the Mac. This guide will get you up and running. Even if you don't intend to edit HTML directly and instead plan to use an HTML editor such as Netscape Composer, or W3C's Amaya, this guide will enable you to understand enough to make better use of such tools and how to make your HTML documents accessible on a wide range of browsers. Once you are comfortable with the basics of authoring HTML, you may want to learn how to add a touch of style using CSS, and to go on to try out features covered in my page on advanced HTML

A convenient way to automatically fix markup errors is to use the HTML Tidy utility. This also tidies the markup making it easier to read and easier to edit. I recommend you regularly run Tidy over any markup you are editing. Tidy is very effective at cleaning up markup created by authoring tools with sloppy habits.

p.s. a good way to learn is to look at how other people have coded their html pages. To do this, click on the "View" menu and then on "Source". Try it with this page to see how I have applied the ideas I explain below. You will find yourself developing a critical eye as many pages look rather a mess under the hood!

This page will teach you how to:

- start with a title
- add headings and paragraphs
- add emphasis to your text
- add images
- add links to other pages
- use various kinds of lists

If you are looking for something else, try the advanced HTML page.

Start with a title

Every HTML document needs a title. Here is what you need to type:

```
<title>My first HTML document</title>
```

Change the text from "My first HTML document" to suit your own needs. The title text is preceded by the start tag <title> and ends with the matching end tag </title>. The title should be placed at the beginning of your document.

To try this out, type the above into a text editor and save the file as "test.html", then view the file in a web browser. If the file extension is ".html" or ".htm" then the browser will recognize it as HTML. Most browsers show the title in the window caption bar.

Add headings and paragraphs

If you have used Microsoft Word, you will be familiar with the built in styles for headings of differing importance. In HTML there are six levels of headings. H1 is the most important, H2 is slightly less important, and so on down to H6, the least important.

Here is how to add an important heading:

```
<h1>An important heading</h1>
```

and here is a slightly less important heading:

```
<h2>A slightly less important heading</h2>
```

Each paragraph you write should start with a `<p>` tag. The `</p>` is optional, unlike the end tags for elements like headings. For example:

```
<p>This is the first paragraph.</p>
<p>This is the second paragraph.</p>
```

Adding a bit of emphasis

You can emphasize one or more words with the `` tag, for instance:

```
This is a really <em>interesting</em> topic!
```

Adding interest to your pages with images

Images can be used to make your Web pages distinctive and greatly help to get your message across. The simple way to add an image is using the `` tag. Let's assume you have an image file called "peter.jpg" in the same folder/directory as your HTML file. It is 200 pixels wide by 150 pixels high.

```

```

The `src` attribute names the image file. The width and height aren't strictly necessary but help to speed the display of your Web page. Something is still missing! People who can't see the image need a description they can read in its absence. You can add a short description as follows:

```

```

The `alt` attribute is used to give the short description, in this case "My friend Peter". For complex images, you may need to also give a longer description. Assuming this has been written in the file "peter.html", you can add one as follows using the `longdesc` attribute:

```

```

You can create images in a number of ways, for instance with a digital camera, by scanning an image in, or creating one with a painting or drawing program. Most browsers understand GIF and JPEG image formats, newer browsers also understand the PNG image format. To avoid long delays while the image is downloaded over the network, you should avoid using large image files.

Generally speaking, JPEG is best for photographs and other smoothly varying images, while GIF and PNG are good for graphics art involving flat areas of color, lines and text. All three formats support options for progressive rendering where a crude version of the image is sent first and progressively refined.

Adding links to other pages

What makes the Web so effective is the ability to define links from one page to another, and to follow links at the click of a button. A single click can take you right across the world!

Links are defined with the `<a>` tag. Lets define a link to the page defined in the file "peter.html":

```
This a link to <a href="peter.html">Peter's page</a>.
```

The text between the `<a>` and the `` is used as the caption for the link. It is common for the caption to be in blue underlined text.

To link to a page on another Web site you need to give the full Web address (commonly called a URL), for instance to link to www.w3.org you need to write:

```
This is a link to <a href="http://www.w3.org/">W3C</a>.
```

You can turn an image into a hypertext link, for example, the following allows you to click on the company logo to get to the home page:

```
<a href="/"></a>
```

Three kinds of lists

HTML supports three kinds of lists. The first kind is a bulleted list, often called an *unordered list*. It uses the `` and `` tags, for instance:

```
<ul>
  <li>the first list item</li>
  <li>the second list item</li>
  <li>the third list item</li>
</ul>
```

Note that you always need to end the list with the `` end tag, but that the `` is optional and can be left off. The second kind of list is a numbered list, often called an *ordered list*. It uses the `` and `` tags. For instance:

```
<ol>
  <li>the first list item</li>
  <li>the second list item</li>
  <li>the third list item</li>
</ol>
```

Like bulleted lists, you always need to end the list with the `` end tag, but the `` end tag is optional and can be left off.

The third and final kind of list is the definition list. This allows you to list terms and their definitions. This kind of list starts with a `<dl>` tag and ends with `</dl>`. Each term starts with a `<dt>` tag and each definition starts with a `<dd>`. For instance:

```
<dl>
  <dt>the first term</dt>
  <dd>its definition</dd>
  <dt>the second term</dt>
  <dd>its definition</dd>
  <dt>the third term</dt>
  <dd>its definition</dd>
</dl>
```

The end tags `</dt>` and `</dd>` are optional and can be left off. Note that lists can be nested, one within another. For instance:

```
<ol>
  <li>the first list item</li>
  <li>
    the second list item
    <ul>
      <li>first nested item</li>
      <li>second nested item</li>
    </ul>
  </li>
  <li>the third list item</li>
</ol>
```

You can also make use of paragraphs and headings etc. for longer list items.

Getting Further Information

If you are ready to learn more, I have prepared some accompanying material on advanced HTML and adding a touch of style.

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>

Copyright © 1994-2003 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply. Your interactions with this site are in accordance with our public and Member privacy statements.



More advanced features

Dave Raggett, 29th August 2000.

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

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&nbsp;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 ` `

```
<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>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>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>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:

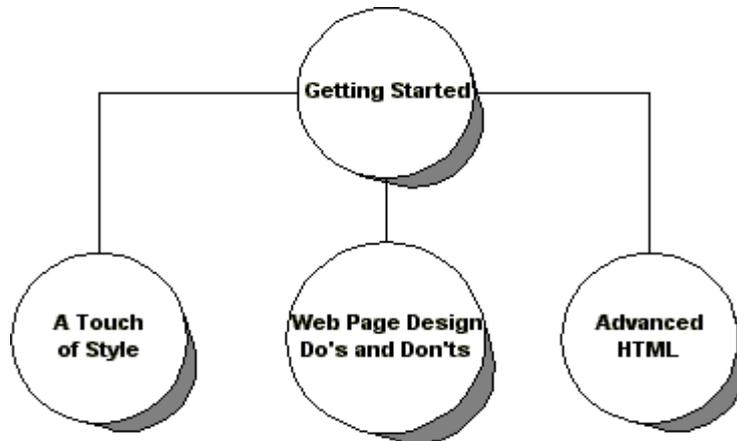


This text will be flowed around the right side of the graphic.

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">
  
  <map name="sitemap">
    <area shape="circle" coords="186,44,45"
      href="guide.htm" alt="Getting Started">
    <area shape="circle" coords="42,171,45"
      href="style.htm" 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.htm" 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: $x_1, y_1, x_2, y_2, \dots, x_n, y_n$

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 annular 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="guide.htm" 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 width="50%">Year</th><th width="50%">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 width="50%">Year</th><th width="50%">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>&nbsp;</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" width="25%">Year</th>
<th colspan="3" width="75%">Sales</th></tr>
<tr align="center"><th width="25%">North</th><th width="25%">South</th>
<th width="25%">Total</th></tr>
<tr align="center"><td width="25%">2000</td><
td width="25%">$10M</td><td width="25%">$8M</td><td width="25%">$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 width="25%">North<th width="25%">South<th width="25%">Total
<tr align="center"><td width="25%">2000
<td width="25%">$10M<td width="25%">$8M<td width="25%">$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 width="50%">Year</th><th width="50%">Sales</th></tr>
```

```
|<td>2000</td><td>$18M</td></tr>
|<td>2001</td><td>$25M</td></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" width="50%">Year</th>
    <th bgcolor="#CCCC99" width="50%">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>

```

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 width="50%">Year</th><th width="50%">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:

Projected sales revenue by year

Year	Sales
2000	\$18M
2001	\$25M

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" width="50%">Year</th>
  <th scope="col" width="50%">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 appearance 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 appearance 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")'></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/mit.gif",
                          "hosts/inria.gif", "hosts/keio.gif");
    adURLs = new Array("www.lcs.mit.edu",
                       "www.inria.fr", "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()"></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.

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>

Copyright © 1994-2003 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply. Your interactions with this site are in accordance with our public and Member privacy statements.



Adding a touch of style

Dave Raggett, 8th April 2002.

This is a short guide to styling your Web pages. It will show you how to use W3C's Cascading Style Sheets language (CSS) as well as alternatives using HTML itself. The route will steer you clear of most of the problems caused by differences between different brands and versions of browsers.

For style sheets to work, it is important that your markup be free of errors. A convenient way to automatically fix markup errors is to use the HTML Tidy utility. This also tidies the markup making it easier to read and easier to edit. I recommend you regularly run Tidy over any markup you are editing. Tidy is very effective at cleaning up markup created by authoring tools with sloppy habits.

The following will teach you how to:

- use the style element
- link to separate style sheets
- set page margins
- set left and right and first-line indents
- set the amount of whitespace above and below
- set the font type, style and size
- add borders and backgrounds
- set colors with named or numeric values
- add style for browsers that don't understand CSS

Getting started

Let's start with setting the color of the text and the background. You can do this by using the STYLE element to set style properties for the document's tags:

```
<style type="text/css">
  body { color: black; background: white; }
</style>
```

The stuff between the <style> and </style> is written in special notation for style rules. Each rule starts with a tag name followed by a list of style properties bracketed by { and }. In this example, the rule matches the **body** tag. As you will see, the body tag provides the basis for setting the overall look and feel of your Web page.

Each style property starts with the property's name, then a colon and lastly the value for this property. When there is more than one style property in the list, you need to use a semicolon between each of them to delimit one property from the next. In this example, there are two properties - "color" which sets the color of the text, and "background" which sets the color of the page background. I recommend always adding the semicolon even after the last property.

Colors can be given as names or as numerical values, for instance `rgb(255, 204, 204)` which is a fleshy pink. The 3 numbers correspond to red, green and blue respectively in the range 0 to 255. You can also use a hexadecimal notation, the same color can also be written as `#FFCCCC`. More details on color are given in a later section.

Note that the style element must be placed in the document's head along with the title element. It shouldn't be placed within the body.

Linking to a separate style sheet

If you are likely to want to use the same styles for several Web pages it is worth considering using a separate style sheet which you then link from each page.

You can do this as follows:

```
<link type="text/css" rel="stylesheet" href="style.css">
```

The LINK tag should be placed in the document's head. The *rel* attribute must be set to the value "stylesheet" to allow the browser to recognize that the *href* attribute gives the Web address (URL) for your style sheet.

Setting the page margins

Web pages look a lot nicer with bigger margins. You can set the left and right margins with the "margin-left" and "margin-right" properties, e.g.

```
<style type="text/css">
  body { margin-left: 10%; margin-right: 10%; }
</style>
```

This sets both margins to 10% of the window width, and the margins will scale when you resize the browser window.

Setting left and right indents

To make headings a little more distinctive, you can make them start within the margin set for the body, e.g.

```
<style type="text/css">
  body { margin-left: 10%; margin-right: 10%; }
  h1 { margin-left: -8%;}
  h2,h3,h4,h5,h6 { margin-left: -4%; }
</style>
```

This example has three style rules. One for the body, one for h1 (used for the most important headings) and one for the rest of the headings (h2, h3, h4, h5 and h6). The margins for the headings are additive to the margins for the body. Negative values are used to move the start of the headings to the left of the margin set for the body.

In the following sections, the examples of particular style rules will need to be placed within the style element in the document's head (if present) or in a linked style sheet.

Controlling the white space above and below

Browsers do a pretty good job for the white space above and below headings and paragraphs etc. Two reasons for taking control of this yourself are: when you want a lot of white space before a particular heading or paragraph, or when you need precise control for the general spacings.

The "margin-top" property specifies the space above and the "margin-bottom" specifies the space below. To set these for all h2 headings you can write:

```
h2 { margin-top: 8em; margin-bottom: 3em; }
```

The em is a very useful unit as it scales with the size of the font. One em is the height of the font. By using em's you can preserve the general look of the Web page independently of the font size. This is much safer than alternatives such as pixels or points, which can cause problems for users who need large fonts to read the text.

Points are commonly used in word processing packages, e.g. 10pt text. Unfortunately the same point size is rendered differently on different browsers. What works fine for one browser will be illegible on another! Sticking with em's avoids these problems.

To specify the space above a particular heading, you should create a named style for the heading. You do this with the *class* attribute in the markup, e.g.

```
<h2 class="subsection">Getting started</h2>
```

The style rule is then written as:

```
h2.subsection { margin-top: 8em; margin-bottom: 3em; }
```

The rule starts with the tag name, a dot and then the value of the class attribute. Be careful to avoid placing a space before or after the dot. If you do the rule won't work. There are other ways to set the styles for a particular element but the class attribute is the most flexible.

When a heading is followed by a paragraph, the value for margin-bottom for the heading isn't added to the value for margin-top for the paragraph. Instead, the maximum of the two values is used for the spacing between the heading and paragraph. This subtlety applies to margin-top and margin-bottom regardless of which tags are involved.

First-line indent

Sometimes you may want to indent the first line of each paragraph. The following style rule emulates the traditional way paragraphs are rendered in novels:

```
p { text-indent: 2em; margin-top: 0; margin-bottom: 0; }
```

It indents the first line of each paragraph by 2 em's and suppresses the inter-paragraph spacing.

Controlling the font

This section explains how to set the font and size, and how to add italic, bold and other styles.

Font styles

The most common styles are to place text in italic or bold. Most browsers render the **em** tag in italic and the **strong** tag in bold. Let's assume you instead want em to appear in **bold italic** and strong in **bold uppercase**:

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

If you feel so inclined, you can fold headings to lower case as follows:

```
h2 { text-transform: lowercase; }
```

Setting the font size

Most browsers use a larger font size for more important headings. If you override the default size, you run the risk of making the text too small to be legible, particularly if you use points. You are therefore recommended to specify font sizes in relative terms.

This example sets heading sizes in percentages relative to the size used for normal text:

```
h1 { font-size: 200%; }
h2 { font-size: 150%; }
h3 { font-size: 100%; }
```

Setting the font family

It is likely that your favorite font won't be available on all browsers. To get around this, you are allowed to list several fonts in preference order. There is a short list of generic font names which are guaranteed to be available, so you are recommended to end your list with one of these: serif, sans-serif, cursive, fantasy, or monospace, for instance:

```
body { font-family: Verdana, sans-serif; }
h1,h2 { font-family: Garamond, "Times New Roman", serif; }
```

In this example, important headings would preferably be shown in Garamond, failing that in Times New Roman, and if that is unavailable in the browsers default serif font. Paragraph text would appear in Verdana or if that is unavailable in the browser's default sans-serif font.

The legibility of different fonts generally depends more on the height of lower case letters than on the font size itself. Fonts like Verdana are much more legible than ones like "Times New Roman" and are therefore recommended for paragraph text.

Avoid problems with fonts and margins

My first rule is to avoid text at the body level that isn't wrapped in a block level element such as **p**. For instance:

```
<h2>Spring in Wiltshire</h2>
Blossom on the trees, bird song and the sound of lambs
bleating in the fields.
```

The text following the heading runs the risk on some browsers of being rendered with the wrong font and margins. You are therefore advised to enclose all such text in a paragraph, e.g.

```
<h2>Spring in Wiltshire</h2>
<p>Blossom on the trees, bird song and the sound of lambs
bleating in the fields.</p>
```

My second rule is to set the font family for **pre** elements, as some browsers forget to use a fixed pitch font when you set the font size or other properties for pre.

```
pre { font-family: monospace; }
```

My third rule is to set the font family on headings, p and ul elements if you intend to set borders or backgrounds on elements such as div. This is a work-around for a bug where the browser forgets to use the inherited font family, instead switching to the default font as set by the browser preferences.

```
h1,h2,h3,h4,h5,p,ul { font-family: sans-serif; }
```

Adding borders and backgrounds

You can easily add a border around a heading, list, paragraph or a group of these enclosed with a **div** element. For instance:

```
div.box { border: solid; border-width: thin; width: 100% }
```

Note that without the "width" property some browsers will place the right margin too far to the right. This can then be used with markup such as:

```
<div class="box">
  The content within this DIV element will be enclosed
  in a box with a thin line around it.
</div>
```

There are a limited choice of border types: dotted, dashed, solid, double, groove, ridge, inset and outset. The border-width property sets the width. Its values include thin, medium and thick as well as a specified width e.g. 0.1em. The border-color property allows you to set the color.

A nice effect is to paint the background of the box with a solid color or with a tiled image. To do this you use the background property. You can fill the box enclosing a div as follows:

```
div.color {
  background: rgb(204,204,255);
  padding: 0.5em;
  border: none;
}
```

Without an explicit definition for border property some browsers will only paint the background color under each character. The padding property introduces some space between the edges of the colored region and the text it contains.

You can set different values for padding on the left, top, right and bottom sides with the padding-left, padding-top, padding-right and padding-bottom properties, e.g. `padding-left: 1em`.

Suppose you only want borders on some of the sides. You can control the border properties for each of the sides independently using the border-left, border-top, border-right and border-bottom family of properties together with the appropriate suffix: style, width or color, e.g.

```
p.changed {
  padding-left: 0.2em;
  border-left: solid;
  border-right: none;
  border-top: none;
  border-bottom: none;
  border-left-width: thin;
  border-color: red;
}
```

which sets a red border down the left hand side only of any paragraph with the class "changed".

Setting Colors

Some examples for setting colors appeared in earlier sections. Here is a reminder:

```
body {
  color: black;
```

```

background: white;
}
strong { color: red }

```

This sets the default to black text on a white background, but renders strong elements in red. There are 16 standard color name, which are explained just below. You can also use decimal values for red, green and blue, where each value appears in the range 0 to 255, e.g. *rgb(255, 0, 0)* is the same as *red*. You can also use hex color values which start with the '#' character followed by six hexadecimal digits. A two-way converter is included below which allows you to convert from RGB to hex color values.

Setting Link Colors

You can use CSS to set the color for hypertext links, with a different color for links that you have yet to follow, ones you have followed, and the active color for when the link is being clicked. You can even set the color for when the mouse pointer is hovering over the link.

```

:link { color: rgb(0, 0, 153) } /* for unvisited links */
:visited { color: rgb(153, 0, 153) } /* for visited links */
:active { color: rgb(255, 0, 102) } /* when link is clicked */
:hover { color: rgb(0, 96, 255) } /* when mouse is over link */

```

Sometimes you may want to show hypertext links without them being underlined. You can do this by setting the *text-decoration* property to *none*, for example:

```
a.plain { text-decoration: none }
```

Which would suppress underlining for a link such as:

```
This is <a class="plain" href="http://www.w3.org/what.html">not underlined</a>
```

Most people when they see underlined text on a Web page, will expect it to be part of a hypertext link. As a result, you are advised to leave underlining on for hypertext links. A similar argument applies to the link colors, most people will interpret underlined blue text as hypertext links. You are advised to leave link colors alone, except when the color of the background would otherwise make the text hard to read.

Color Blindness

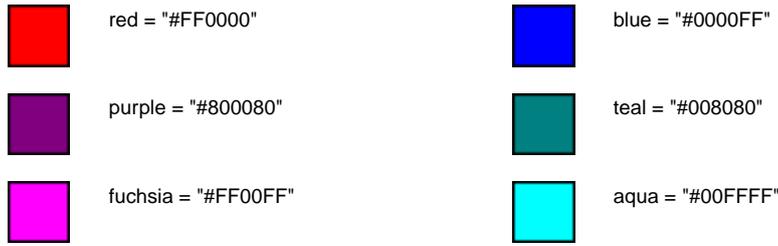
When using color, remember that 5 to 10% of men have some form of color blindness. This can cause difficulties distinguishing between red and green, or between yellow and blue. In rare cases, there is an inability to perceive any colors at all. You are recommended to avoid foreground/background color combinations that would make the text hard to read for people with color blindness.

Named colors

The standard set of color names is: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. These 16 colors are defined in HTML 3.2 and 4.01 and correspond to the basic VGA set on PCs. Most browsers accept a wider set of color names but use of these is not recommended.

Color names and sRGB values

	black = "#000000"		green = "#008000"
	silver = "#C0C0C0"		lime = "#00FF00"
	gray = "#808080"		olive = "#808000"
	white = "#FFFFFF"		yellow = "#FFFF00"
	maroon = "#800000"		navy = "#000080"



Thus, the color value "#800080" is the same as "purple".

Hexadecimal color values

Values like "#FF9999" represent colors as hexadecimal numbers for red, green and blue. The first two characters following the # give the number for red, the next two for green and the last two for blue. These numbers are always in the range 0 to 255 decimal. If you know the values in decimal, you can convert to hexadecimal using a calculator, like the one that comes as part of Microsoft Windows.

Enter RGB or Hex value and press appropriate button to convert

red:	255	Hex color value
green:	255	
blue:	255	
		#FFFFFF

Browser safe colors

New computers support thousands or millions of colors, but many older color systems can only show up to 256 colors at a time. To cope with this, these browsers make do with colors from a fixed palette. The effect of this is often visible as a speckling of colors as the browser tries to approximate the true color at any point in the image. This problem will gradually go away as older computers are replaced by newer models.

Most browsers support the same so called "browser safe" palette. This uses 6 evenly spaced gradations in red, green and blue and their combinations. If you select image colors from this palette, you can avoid the speckling effect. This is particularly useful for background areas of images.

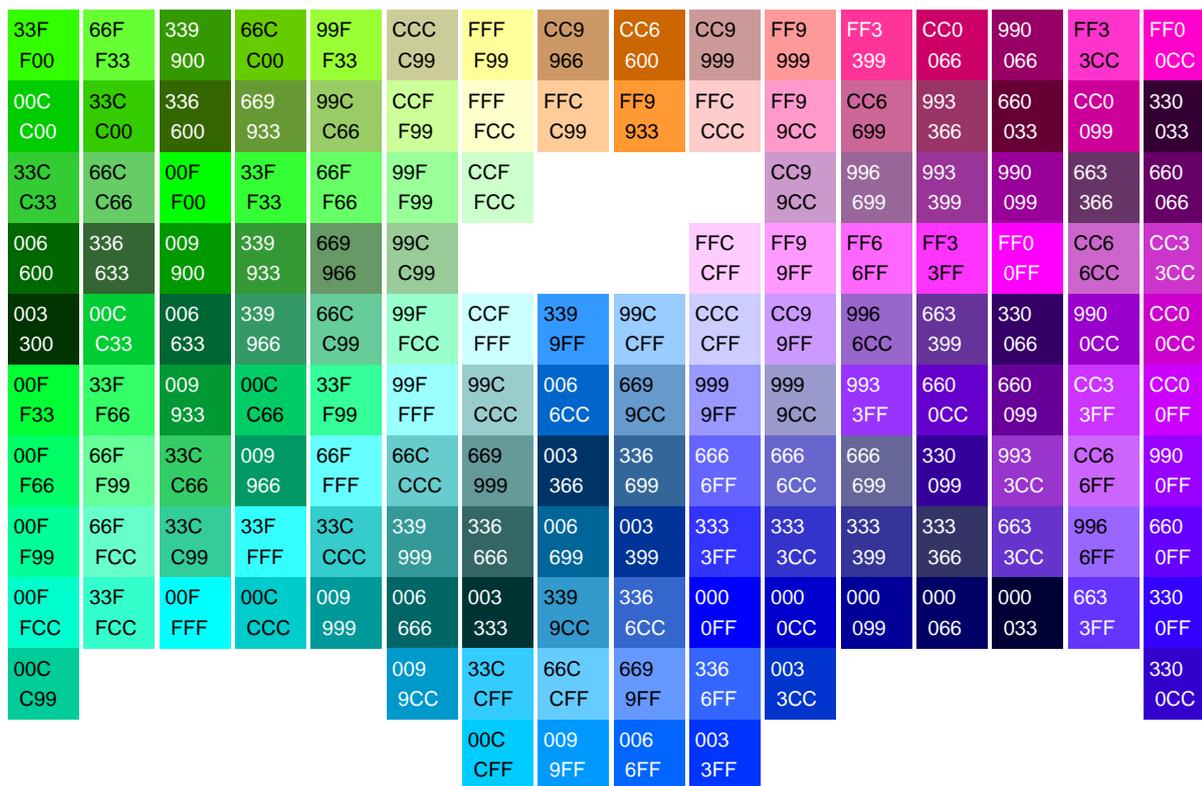
If the browser is using the browser safe palette, the page background uses the nearest color in the palette. If you set the page background to a color which isn't in the browser safe palette, you run the risk that the background will have different colors depending on whether the computer is using indexed or true-color.

These are constructed from colors where red, green and blue are restricted to the values:

RGB	00	51	102	153	204	255
Hex	00	33	66	99	CC	FF

Here is a table of the browser safe colors and their hex values. My thanks to Bob Stein for this arrangement.

FFF	CCC	999	666	333	000	FFC	FF9	FF6	FF3							
FFF	CCC	999	666	333	000	C00	900	600	300							
99C					CC9	FFC	FFC	FF9	FF6	CC3						CC0
C00					900	C33	C66	966	633	300						033
CCF	CCF	333	666	999	CCC	FFF	CC9	CC6	330	660	990	CC0	FF0	FF3	FF0	
F00	F33	300	600	900	C00	F00	933	633	000	000	000	000	000	366	033	
99F	CCF	99C	666	999	CCC	FFF	996	993	663	993	CC3	FF3	CC3	FF6	FF0	
F00	F66	C33	633	933	C33	F33	600	300	333	333	333	333	366	699	066	
66F	99F	66C	669	999	CCC	FFF	996	663	996	CC6	FF6	990	CC3	FF6	FF0	
F00	F66	C33	900	966	C66	F66	633	300	666	666	666	033	399	6CC	099	



Color swatches for the browser safe palette are available free for many popular graphics packages, from www.visibone.com.

What about browsers that don't support CSS?

Older browsers, that is to say before Netscape 4.0 and Internet Explorer 4.0, either don't support CSS at all or do so inconsistently. For these browsers you can still control the style by using HTML itself.

Setting the color and background

You can set the color using the BODY tag. The following example sets the background color to white and the text color to black:

```
<body bgcolor="white" text="black">
```

The BODY element should be placed before the visible content of the Web page, e.g. before the first heading. You can also control the color of hypertext links. There are three attributes for this:

- **link** for unvisited links
- **vlink** for visited links
- **alink** for the color used when you click the link

Here is an example that sets all three:

```
<body bgcolor="white" text="black"
  link="navy" vlink="maroon" alink="red">
```

You can also get the browser to tile the page background with an image using the background attribute to specify the Web address for the image, e.g.

```
<body bgcolor="white" background="texture.jpeg" text="black"
  link="navy" vlink="maroon" alink="red">
```

It is a good idea to specify a background color using the bgcolor attribute in case the browser is unable to render the image. You should check that the colors you have chosen don't cause legibility problems. As an extreme case consider the following:

```
<body bgcolor="black">
```

Most browsers will render text in black by default. The end result is that the page will be shown with black text on a black background! Lots of people suffer from one form of color blindness or another, for example olive green may appear brown to some people.

A separate problem appears when you try to print the Web page. Many browsers will ignore the background color, but will obey the text color. Setting the text to white will often result in a blank page when printed, so the following is not recommended:

```
<body bgcolor="black" text="white">
```

You can also use the bgcolor attribute on table cells, e.g.

```
<table border="0" cellpadding="5">
  <tr>
    <td bgcolor="yellow">colored table cell</td>
  </tr>
</table>
```

Tables can be used for a variety of layout effects and have been widely exploited for this. In the future this role is likely to be supplanted by style sheets, which make it practical to achieve precise layout with less effort.

Setting the font, its size and color

The FONT tag can be used to select the font, to set its size and the color. This example just sets the color:

```
This sentence has a <font color="yellow">word</font> in yellow.
```

The **face** attribute is used to set the font. It takes a list of fonts in preference order, e.g.

```
<font face="Garamond, Times New Roman">some text ...</font>
```

The **size** attribute can be used to select the font size as a number from 1 to 6. If you place a - or + sign before the number it is interpreted as a relative value. Use size="+1" when you want to use the next larger font size and size="-1" when you want to use the next smaller font size, e.g.

```
<font size="+1" color="maroon"
  face="Garamond, Times New Roman">some text ...</font>
```

There are a couple of things you should avoid: Don't choose color combinations that make text hard to read for people who are color blind. Don't use font to make regular text into headings, which should always be marked up using the h1 to h6 tags as appropriate to the importance of the heading.

Getting Further Information

For further information on CSS and tools that support it, you should look at the W3C home page for CSS. This includes pointers to books on HTML and CSS, for example, "Raggett on HTML 4", published 1998 by Addison Wesley. For a more detailed explanation of CSS, "Cascading Style Sheets" by Håkon Wium Lie and Bert Bos, pub. 1999 by Addison Wesley, which provides an in-depth look at CSS as seen by the architects of CSS themselves.

I plan to extend this guide with additional pages explaining CSS positioning, printing and aural style sheets.

Best of luck and get writing!

Copyright © 1994-2003 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply. Your interactions with this site are in accordance with our public and Member privacy statements.