

## 08 Formatting With Style

### 1 Introduction to Styles

Anyone who's used HTML for a while knows something of its limitations. HTML was never originally designed to look good. For years, web designers have been getting around this problem with creative use of tables, graphics etc. These techniques work, but they are awkward, hard to modify and generally produce bloated pages that aren't at all modern friendly.

Late in 1996 a new standard called *cascading stylesheets* was proposed. This was incorporated into version 4.0 of the HTML standard. It was a little slow to catch on but it has now gained acceptance among the major browser vendors, to the point where the majority of web browsers now recognize it. The name comes from the fact that specified formatting can be cascaded to many parts of a website.

HTML originally defined the structure of the document but left the formatting up to the browser. E.g. <H1> meant a major heading but how that heading looked was up to the browser. Tags such as <FONT> and <B> were later added to define the look of a document. Stylesheets are intended to handle the formatting of the site and to keep the formatting separate from the HTML, so that the HTML can be used for the structure of the site as it was originally intended. Furthermore, this can all be done from the one place so that instead of defining the formatting for each heading, we only need to place a single line that specifies how all of our headings will look. This means less HTML code, more consistency and easier modifications. Below are some of the advantages of using stylesheets:

- More precise control than ever before over layout, fonts, colours, backgrounds, and other typographical effects.
- A way to update the appearance and formatting of an unlimited number of pages by changing just one document.
- Compatibility across browsers and platforms.
- Less code, smaller pages, and faster downloads.

Of course, older browsers won't recognize stylesheets but that's not a huge problem. Like any HTML, browsers that recognize it, use it. Browsers that don't recognize it, ignore it so that it won't cause problems in old browsers (often referred to as "degrading gracefully").

Like most HTML, there are differences in the way different browsers interpret stylesheets. Internet Explorer 3 was the first browser to support stylesheets. The stylesheet support was incomplete, however, since the standards had yet to be finalised. Internet Explorer 4 and Navigator 4 support stylesheets in a more complete way, though there are still differences in the way they support it. Most things work but you may find a few elements of stylesheets that won't work in both browsers. The latest versions of Internet Explorer (version 6) and Netscape Navigator (Version 7) contain fairly complete support for the stylesheet specification.

### 2 Structure of Styles

The basic elements of styles are rules, declarations and selectors. An example of a rule could be

```
H1 {color: red}
```

This rule tells the web browser that level 1 headings should have a red font colour. Within the rule are the selector H1 (can be any HTML tag), which tells the browser what the style is to be applied to (level 1 heading tags in this case) and the declaration that follows. The declaration specifies how the selector will be displayed/formatted. The declaration can contain several elements, described later on page 6.

## 2.1 Grouping selectors

If you have several HTML tags that you want formatted in a similar way you can create a style rule for all of them together. For example, if you wanted your main heading sizes and paragraphs to all have navy blue text, instead of creating a style rule for each one, you could group them together as in the following example.

```
H1, H2, H3, P {color: navy}
```

Doing this saves space and simplifies any future changes.

## 3 Defining Styles

There are four ways you can define styles.

Linked stylesheets

Embedded stylesheets

Imported Stylesheets

Inline Styles

### 3.1 Linked Style Sheets

In this method, you first create a document with a .css extension that will contain the style rules you want to use throughout your site. The document doesn't need to contain anything else. You then put a line in each document that will link to your css file. Below is an example of what might be contained in a css file:

```
H1 {color: red; font-family: Arial}
P   {color: navy; font-family: Times New Roman}
```

We'll assume that this file was saved with the name mystyle.css. To use this stylesheet file in our website, we would need to put the following line in the <HEAD> section of each page:

```
<LINK REL="stylesheet" HREF="mystyle.css" TYPE="text/css">
```

This would tell the browser to use the styles defined in that external file. The great advantage of this method of using styles is that your styles only need to be defined once. In the above example, we have specified that level 1 headings are red and Arial. This will be used in every document that links to mystyle.css. To do this without styles, you would need to go to every page in your site, find every <H1> tag, and put in the appropriate <FONT> tag. Not only does this mean more HTML but it also creates a real headache if you later decide you want to change the way your level 1 headings look. With a linked stylesheet, you make the change once and that change is reflected throughout your site.

## 3.2 Embedded stylesheets

With an embedded stylesheet, you define your styles in a <STYLE> block at the top of an HTML document. These style rules are only used within that document. The style block is placed in the <HEAD> section of the page as in the following example:

```
<HTML>
<HEAD>
<STYLE TYPE="TEXT/CSS">
<!--
H1 {color: red}
-->
</STYLE>
</HEAD>
<BODY>

</BODY>
</HTML>
```

In this example all of the level 1 headings within the document will be red but other documents won't be affected. You may have noticed some unusual HTML and some comments. The TEXT/CSS part specifies the type of code used so browsers that don't recognise it can ignore it. Some older browsers still won't recognise that, so the comment tags prevent them from displaying the html text on the page itself.

## 3.3 Imported stylesheets

This is like combining embedded and linked stylesheets. This can be used to define styles within a particular page, while at the same time importing styles from a css file. It also allows you to import style rules from more than one css file. When we create the inline style at the top of the page, we include a line that will import the styles from our external stylesheet as in the following example:

```
<STYLE TYPE="TEXT/CSS">
<!--
@import url(mystyle.css)
H1 {font-family: Arial; color: red}
-->
</STYLE>
```

<b>Note</b> Netscape Navigator 4 does not support this method
---

## 3.4 Inline styles

With inline styles you can specify the style for a particular part of the page, a bit like the way you would with a <FONT> tag. The difference is that styles allow a lot more control than older HTML such as the <FONT> tag as we will see later. If we wanted a particular paragraph to take on certain formatting, rather than using one of the older methods to change all our paragraph tags, we could use an inline style to change only the one we want by inserting a STYLE= attribute to our <P> tag. This is demonstrated in the following example.

```
<P STYLE="color: red">
```

As you can see, inline styles are inserted in tags the same as other attributes in HTML and they can go in pretty much any HTML tag. Inline styles are the only case where a style declaration is not surrounded by curly brackets. { }

<b>Note</b> There are some cases where inline styles might not work too well, E.g. some browsers don't correctly handle inline styles in image tags, though most tags are fine.
---

### 3.5 Order of inheritance

Often there is a conflict between style definitions. E.g. an imported stylesheet says headings are red, the embedded style block says they're blue and an inline style says a particular heading is green. In this case, that one heading would be green, every other heading on the page would be blue and every heading on other pages would be red. This is because inline styles override embedded styles, which override linked/imported styles.



If there is no clear preference (e.g. Two P sections in a css file), the browser will usually use the last one specified.

Where a stylesheets conflicts with HTML, we have a problem. E.g. if a style says text should be Arial but a <FONT> tag says it should be Times New Roman, what happens? According to the HTML specification, browsers that recognise styles should use them while other browsers will use the <FONT> tag. Unfortunately, Netscape Navigator 4 and Internet Explorer 4 currently use HTML tags over conflicting style tags. Internet Explorer 5 and Navigator 6 correctly give the style rules precedence over older <FONT> tags.

## 4 Stylesheet Classes

Stylesheets give you the capability to specify different formatting for the same HTML tag. For example, if you wanted some of your paragraphs to be one colour and other paragraphs to be a different colour. Suppose we had a document that contained the transcript of an interview where the text for what the host said needed to be navy blue while the text for the guest needed to be green. We could use classes in our styles to specify different formatting for different paragraphs as shown below. The classes are defined by putting the name of the class after the HTML tag name with a full stop between them.

```
P.host {color: navy}
P.guest {color: green}
```

This defines one paragraph style for the host of the interview and one for the guest in the interview. We can then use a CLASS= attribute to specify where they are to be used within the document. The CLASS attribute specifies which Style class will be used for that tag as in the following example.

```
<P CLASS="host">Interviewer text formatted in navy blue</P>
<P CLASS="guest">Guest text formatted in navy green</P>
```

If the colour were changed later, it would only need to be changed in the style definition, not in each paragraph. It also uses a lot less HTML than if you had to place <FONT> tags all the way down the page, especially if you are doing more than just changing colours. The name of the class is up to you. Just make sure you include the full stop between the tag name and the class name.

You don't actually have to have a class assigned to a particular tag. If you prefer, you can have a more generic class that can be used in any tag. For example, we could have a class for any news items such as the one below:

```
.news {color: red}
```

We could use this in any tag. This would mean that any tag could make use of this class and it's associated formatting. For example, below, we will use our "news" class to format a heading and a paragraph:

```
<H2 CLASS="news">news heading</H2>
<P CLASS="news">News content</P>
```

### 4.1 Contextual selectors

It is possible to specify that a style will only be used in certain places. For example, you can use a style to specify that any text formatted with a <B> tag will be red, but only if it's part of a level 1 heading. To do this, we would create a style rule with the selector as H1 followed by B as shown below. (Note that there is no full stop or comma between the H1 and B tags)

```
H1 B {color: red}
```

The result of this would be that any bold text that is part of a level 1 heading would be red.

### 4.2 Style comments

You can place comments in a style block or style sheet that will be ignored by any web browser. To place a comment, insert the text between a /\* and a \*/. E.g.

```
P {color: green} /* this style makes paragraph text green */
```

## 5 Some Stylesheet Attributes

By now you would have seen some stylesheet attributes being used in the examples that we have looked at. In this section we will look at some of the many attributes available. These attributes may be placed in the style declaration for inline styles, embedded styles and external (css) stylesheets. Some of these attributes with explanations are listed below. Note that some of them (such as margin and text-align) only work on block HTML tags (ones that define their own paragraph) such as <P> <DIV> <H1> and <LI>. This isn't a complete list but others in the official specification aren't correctly supported in many current browsers.

background	Sets the background color for the formatted area. This works similar to the BGCOLOR attribute in a <BODY> tag except that this can be used for a wide variety of tags. Colours can be set using a colour name eg color: red or a hexadecimal value eg color: #FF0000. Unlike the BGCOLOR attribute tag you can also specify colour as an RGB value eg color: rgb(255,0,0)
background-image	Uses an image as the background for the formatted area. E.g. background-image: url(back.gif) Like the background attribute, this can be used for more than just the page body so you can set background graphics for individual paragraphs and other tags.
color	Like the <FONT> tag, this attribute allows you to set the foreground colour of text. Colours are set the same as background colours (see above)..
font-family	Sets the font to be used (eg arial). Like the font tag, you can list more than one font separated by commas. If the font name is not one word (eg Comic Sans MS) it may help to surround the name with double quotes or single quotes for an inline style.
font-size	Rather than being stuck with the <FONT> tag's 7 sizes, this attribute allows you to be more specific. There are several ways font size can be set. One is by using units such as point size (pt), pixels (px), inches (in), centimetres (cm), millimetres (mm) and picas (pc). For example font-size: 1.2cm would make the font 1.2cm tall. Another way to set the size is with keywords such as <u>xx-small</u> , <u>x-small</u> , <u>small</u> , <u>medium</u> , <u>large</u> , <u>x-large</u> , <u>xx-large</u> , <u>smaller</u> and <u>larger</u> . The third way to set font size is with percentages and decimals. For example font-size: 150% would make the font 1.5 times the normal size.
font-style	This attribute can be set to <u>italic</u> , <u>oblique</u> (sometimes used instead of italic) or <u>normal</u> (normal cancels out any italic effect inherited from elsewhere).
font-transform	This controls the capitalisation of the text with 4 choices. <u>uppercase</u> CONVERTS ALL THE TEXT TO UPPERCASE. <u>lowercase</u> converts all text to lowercase. <u>capitalize</u> Converts The First Letter Of Every Word To Upper Case. <u>none</u> cancels any inherited transform value.
font-weight	The most basic uses of this are to set it to <u>bold</u> or <u>normal</u> . Additionally you can control <u>how</u> bold the text will be using a number from 100 to 900. Regular bold text would be 400. The keywords <u>lighter</u> and <u>bolder</u> may also be used for text that is already been made bold another way.
line-height	Controls the distance between the bottom of one line and the bottom of the next line (line spacing). This can be specified as a number or a percentage. E.g. 2 would mean that the height of the line is 2 x the font height. 150% means that the line height is 150% x the font height. It can also be specified as a particular unit. E.g. 15pt would make the line height 15 points. Note that if you set a small enough value, your lines will overlap.
margin-	Sets the size of the margins around a block. Each side is set with a different attribute. E.g. margin-left: 1cm and margin-top: 1.5cm
text-align	This allows you to set left/right indents as well as control paragraph spacing
text-decoration	Can be set to left, center, right or justify. Text couldn't be justified previously. Most commonly used to control underlines and has the following options. underline ( <u>line below text</u> ), overline (line above text), line-through ( <u>line through text</u> ), blink (text flashes on and off, none (removes any underline).
text-indent	Indents the first line of a paragraph. You can use the same units that are used for font-size (such as px) or a percentage of the line width. If you put a negative value you will get a hanging indent.

## 5.1 Using Style Attributes

To use any of these in a style declaration, put the name of the attribute (eg color) followed by its value (eg blue) with a colon : separating the two. The following example illustrates this.

```
P {color: blue}
P {font-size: 12pt}
```

Instead of putting several attributes on a separate line as they are in this example, it is often easier to group declarations for the same tag as in the following example. Each one needs to be separated with a semi-colon ;

```
P {color: blue; font-size: 12pt}
```

**Caution** Don't get the colon and semi-colon mixed up. The colon is between an attribute and value within a rule. The semi-colon separates different rules.

## 5.2 5.5.2 Tips for hyperlinks

In styles, hyperlinks have a couple of pre-defined classes that can be used in a linked or embedded stylesheet. These are link, active and visited. Below is an example of how these can be used.

```
A:link {color: blue; text-decoration: none; font-family: "arial"}
A:active {color: red; text-decoration: none; font-family: "arial"}
A:visited {color: purple; text-decoration: none; font-family: "arial"}
A:hover {color: red; text-decoration: underline; font-family: "arial"}
```

These would mean that throughout the affected document/s, all links would be blue Arial, active links would be red Arial and visited links would be purple Arial. Note the use of the text-decoration attribute. This sets the decoration to none meaning that the links would have no underlines. Impossible without styles!

The fourth one is the new hover class. This one allows you to specify a different format for when the mouse is over an anchor. In this case, when the mouse is over a link, the colour of the link will change to red and an underline will appear.

**Exercise 6****Practice with Styles**

We'll now put some of this into practice.

- 1 Open the file styles.html. Look at the HTML and preview it in your browser.
- 2 Add the following style declarations (the bold lines) to the style section already placed (in the <HEAD> section).

```
<STYLE TYPE="TEXT/CSS">
<!--
H1 {font-size: 20pt; font-family: "comic sans ms", arial; font-weight: bold; text-align: center;
color: rgb(255,0,0); text-decoration: underline}
H2 {font-size: 15pt; font-family: Arial; font-weight: normal; color: maroon}
P {font-size: 10pt; font-family: Arial; color: #000080}
A {font-size: 10pt; font-family: Arial; text-decoration: none}
-->
</STYLE>
```

- 3 Save and preview the file to see the changes.
- 4 Add the following line to your style block  
P.news {color: yellow; background: red; font-weight: bold}
- 5 Within your document, add a class attribute to the second paragraph tag so that it appears as shown below. (The modified part is formatted in bold text)

```
<P CLASS="news">This paragraph should be formatted using our &quot;news&quot; class</P>
```

- 6 Now we will add an inline style to the third paragraph tag so that it appears as shown below

```
<P STYLE="text-indent: 1cm; color: green">
```

- 7 Save the document and preview it in your browser. It should look similar to the example below.
- 8 Add the following line to your style block.  
**A:hover {text-decoration: underline; color: Red; background-color: #F5DEB3}**
- 9 Preview the document once again and move your mouse over the hyperlink. This line uses the link's hover pseudo class to change the formatting of links when the mouse goes over them.

## Styles Practice

### Using Basic Styles

This exercise will give you some practice at using styles

**This paragraph should be formatted using our "news" class**

We'll use an inline style to format this paragraph. We will also use the text-indent attribute to indent the first line of this paragraph. It will be indented by an amount of one centimetre.

Find out more about using styles.

## 6 Positioning With Styles

With HTML, positioning things on the page can be a real hassle. In the past it was mostly done using tables but styles give new ways of positioning with increased precision. Suppose you wanted a picture to be positioned 150 pixels from the top of the browser window and 200 pixels from the left. You could accomplish this using an inline style (note: for best results, enclose an image in a <DIV> or <SPAN> tag and apply the inline style there. Some browsers don't take kindly to inline styles in images). A <SPAN> tag can be used as a container for inline styles or classes.

You can position anything in your page including text and graphics. To position parts of your page using an inline style, there are four basic attributes that can be used. These are listed below:

left	Distance from the left. Positive values move to the right, negative values move to the left.
position	Specifies whether the positioning is relative or absolute. With absolute positioning, the distances are measured from the left and top of the browser window. With relative positioning, the distances are measured from the position where the section of the page being positioned would normally begin.
top	Distance from the top. Positive values move downward, negative values move upward.
z-index	A value from 1 to 10. When positioned elements overlap, the one with a higher z-index value will appear in front.

### 6.1 Example 1 Absolute positioning

If we wanted to position an image so that it appeared 150 pixels from the top of the browser window and 200 pixels from the left, we could use HTML similar to the HTML shown below.

```
<SPAN STYLE="position: absolute; left: 200px; top: 150px"><IMG SRC="image.gif"></SPAN>
```

### 6.2 Example 2 Relative positioning

If we wanted to position some italicised text so that it was 5 points higher than the rest of the text in the paragraph, we could use the following HTML

```
<P>Normal text in the paragraph. <I STYLE="position: relative; top: -5pt">Italicised text.</I>
```

This would create an effect similar to that shown below

Normal text in the paragraph. *Italicised text.*

### 6.3 Example 3 Overlapping text effects

We could use the relative positioning methods to position two text blocks so that they overlap. The following HTML

```
<H2 STYLE="color: navy; position: relative; left: 5px; top: 10px; z-index: 1">First bit of text</P>
<H2 STYLE="color: red; position: relative; left: 25px; top: -35px; z-index: 10">Second bit of text</P>
```

Would give an effect similar to that shown below.

First bit of text  
Second bit of text

**Exercise 7****Using Styles in a site**

We'll now bring together what we have learned and add styles to the site we have been working on in previous chapters. We will begin by creating a css file for our stylesheet.

- 1 In your HTML editor, create a new file with the style rules shown below. The file can be completely blank apart from these lines.

```
A                {text-decoration: none; font-family: arial}
A.link           {color: blue}
A.active        {color: red}
A.visited       {color: purple}
A:hover         { text-decoration: underline; color: Red}
H3              {font-size: 16pt; font-weight: bold; color: maroon}
P, LI, DT, DD, TD {font-size: 10pt; font-family: Arial; text-align: justified; color: navy}
```

- 2 Save the file as cougar.css

- 3 Add the following line to the <HEAD> section of the file welcome.html

```
<LINK REL=stylesheet HREF="cougar.css" TYPE="text/css">
```

- 4 Add the same line to about.html, events.html, who.html, contacts.html, feedback.html and links.html.

- 5 When all files are saved, preview the file frames.html in your browser. Remember that this is the file that contains our frameset information. Click the links in the navbar to view each page and note what parts have changed from our styles.

- 6 If you take a good look at the HTML you will notice that there is a lot of formatting repeated at the bottom of each page. To simplify and trim our HTML, we'll add two new classes to our style sheet which we'll then reference on each page.

- 7 Open cougar.css and add the following two new lines

```
.question {font-size: 10pt; font-family: Arial; font-weight: bold; text-align: center; color: maroon}
.footer   {font-size: 8pt; font-family: Arial; text-align: right; color: maroon}
```

- 8 The first class will be used to format the second last line on each page. By referring to this class, we can do away with a <FONT> tag, a <B> tag and the ALIGN= attribute. The second class will be used for the last line on each page. On each page, find the following lines at the bottom.

```
<P ALIGN="center">
<FONT SIZE=2 COLOR="maroon" FACE="Arial"><B>
If you have any questions about us, <A HREF="MAILTO:cougar@footy.net.au">Send us an email</A>
</B></FONT>
</P>
<P ALIGN="right">
<FONT SIZE=1 COLOR="maroon" FACE="Arial"><B>
&copy; Cannington Cougars Football Club. Last updated 30<SUP>th</SUP> July 1998
</B></FONT>
</P>
```

Continued on the next page.

9 Change these lines to the following lines

```
<P CLASS="question">
```

```
If you have any questions about us, <A HREF="MAILTO:cougar@footy.net.au">Send us an  
email</A></P>
```

```
<P CLASS="footer">
```

```
&copy; Cannington Cougars Football Club. Last updated 30<SUP>th</SUP> July 1998</P>
```

10 Ensure all pages are saved and preview frames.html in your browser. You probably won't notice any difference, but we're using less HTML – and it only needs to be edited in one place.

11 Finally, we will use styles positioning to place a picture in the top corner of each page. We will begin by inserting the image at the bottom of the page.

12 Open welcome.html and insert the following HTML before the closing </BODY> tag.

```
<A HREF="http://www.afl.com.au" TARGET="_top"><IMG SRC="pics/afl.gif" ALT="Visit The Official  
AFL site" WIDTH="82" HEIGHT="30" BORDER="0"></A>
```

13 We will now place a <SPAN> tag around that image which will be used as a container for an inline style. The inline style will place the graphic at the top of the page. Browsers that don't support styles will leave the picture at the bottom. The HTML should look like the HTML below.

```
<SPAN STYLE="position: absolute; left:85%; top: 0px">
```

```
<A HREF="http://www.afl.com.au" TARGET="_top"><IMG SRC="pics/afl.gif" ALT="Official AFL site"  
WIDTH="82" HEIGHT="30" BORDER="0"></A>
```

```
</SPAN>
```

14 If you have time, copy and paste the new HTML into the other pages, then preview frames.html in your browser.

Previously, doing something as simple as positioning a graphic like this would have required complex tables. It also wouldn't have been as precise.

## 7 Styles Shorthand

In some cases, there may be style declarations that are related to each other such as various font related attributes. If you are using more than one of these then it may be possible to merge them together using styles shorthand. E.g.

```
P {font-style:italic; font-weight:bold; font-size:12pt; font-family:Comic Sans MS, Arial}
```

Could be shortened to:

```
P {FONT: italic bold 12pt Comic Sans MS,Arial}
```

### Exercise 8

### Styles Shorthand

- 1 Open your cougar.css file and modify it so that it looks like the following example.  

```
A {FONT-FAMILY:arial;TEXT-DECORATION:none}
A.ACTIVE {COLOR:red}
A.LINK {COLOR:blue}
A.VISITED {COLOR:purple}
A.HOVER {COLOR:Red;TEXT-DECORATION:underline}
H3 {COLOR:maroon;FONT-SIZE:16pt;FONT-WEIGHT:bold}
P,LI,DT,DD,TD {COLOR:navy;FONT:10pt Arial;TEXT-ALIGN:justified}
.FOOTER {COLOR:maroon;FONT:8pt Arial;TEXT-ALIGN:right}
.QUESTION {COLOR:maroon;FONT:bold 10pt Arial;TEXT-ALIGN:center}
```
- 2 The style rules still perform exactly the same function but they each take up less space now. Note that the spaces between the style selectors and the style declarations are not necessary. They have been added here to make it easier for you to read the style rules.

**Chapter 5****Styles Revision Questions**

1. What are some advantages of using styles instead of other HTML techniques?
2. What can you see are some disadvantages of styles?
3. What are the four ways you can define styles?