Styling and Style Sheets in General

From the beginning, HTML has been a mixture of semantic, stylistic and structural markup. Structural markup was predominant in the early versions of HTML. However, HTML has now developed to the point where appearance can be seen as the more dominant force.

A style sheet is nothing more than a written set of rules laying out the style of a document, and is usually quite separate from the document to be marked up, or at the least is contained in a separate section of the document.

Styling markup on the other hand is markup on the document itself indicating how certain parts of the document are to be styled, and in HTML consists of certain tags and their attributes.

As such, styling markup is quite different from semantic and structural mark up. Structural markup defines the structure of a document, "put a paragraph here, put a heading there", and semantic markup gives information about the content of the document part. It states that "this part of a document is the title, this part is an address" etc.

In this chapter, as a prelude to getting down to a serious examination of style sheets, we will examine the following:

- Styling in HTML, and the complications this can introduce into your coding
- Styling using style sheets, and how this can simplify your coding
- Summarize the advantages of style sheets.

We will do this by looking at how a typical simple page is styled in HTML, and hopefully in the process

the problems associated with HTML styling will become obvious. Just in case they don't we will review them, and then we will look at the potential for style sheets to untangle the mess. We will also take a quick historical look at the reasons behind this mess.

We won't actually look at any of the details of style sheets in this chapter we will leave that to later. First the emphasis will be upon the potential of style sheets to untangle the mess that exists now.

Styling in HTML

Styling in HTML is carried out using a combination of techniques. We will only cover their use very briefly here as books on HTML, such as *Instant HTML Programmer's Reference, ISBN 18610001568 from Wrox Press*, give more than adequate coverage. We will then run through a quick tutorial. If you don't know much about styling using HTML this should be enough for you to get a basic grasp of the subject and, more importantly, make you realize that there must/should be a better way.

First let's look at the various tags and attributes used to style HTML documents.

Styling Tags

These can be divided into tags concerned with paragraph and document layout, and those involved with text and font styles. Also to be considered are attributes of the tags that alter the elements display in some way.

Document Layout

The following table shows the tags most commonly used for document layout.

HTML Tag	Definition
** **	Causes a line break to be inserted into the text. It is also useful for creating horizontal white space in HTML.
<CENTER>	Anything contained between the **<CENTER>** tags will be centered on the page. This tag has no good substitution in style sheets, because the browsers do not implement correctly the properties that are designed to center the elements centering. This means that until the browsers improve the **<CENTER>** tag still has a place in a style-sheet driven document.
<H1><H2> <H3><H4> <H5><H6>	The various headings, display text with a font size that is dependent on the browser, and have a variable amount of white space before or after them.
<HR>	According to many design gurus this should be given a ceremonial burial. Always consider using white space to divide up your page rather than the using horizontal rule.
<P>	The paragraph is the 'work-horse' of HTML. It creates a block of text, usually with a line of white space before and after it.

Text and Font Styling

There are numerous tags that can be used for styling, many of them redundant and seldom used. For a full list see Appendix B of *Instant HTML Programmer's Reference ISBN 18610001568 from Wrox Press* or check out the Ultimate HTML Reference Database at the Reference section of **`http://rapid.wrox.co.uk`**. Here, however, are some of the more commonly used ones. All these tags really act as switches to switch a particular type of styling on or off. Switching on and off style, however, is not a good thought process when we are thinking about styling a page. Instead we need to think of blocks of content with a certain style applied to them. A later paragraph entitled *Switches and Style Blocks* deals with this issue.

HTML Tag	Definition
`<B>`	Turns on **bold** text
`<I>`	Turns on *italic* text
`<U>`	Turns on underlined text
`<S>`	Turns on ~~strikethrough~~ text
`<EM>`	Designed to emphasize text. The text is usually rendered in *italic*.
`<PRE>`	Maintains all the original formatting. Type is usually monospaced.
`<FONT>`	Sets font face, size and color. This is the most powerful of all the HTML tags and as such deserves a subheading of its own.

The FONT Element

The font element is used to switch font characteristics on and off. Among its attributes it has:

Size

This takes the form **`size=`*`n`***, where n can be any number from 1 to 7. **`SIZE=3`** corresponds to a font size used in the main text and **`7`** to an H1 font size. The font can also be made larger by using, for example, **`SIZE=+2`** which would make the font size two sizes bigger, or **`SIZE=-1`** which would make it one size smaller.

Face

This specifies the font family you want to use. It takes the form:

```
FACE= "fontname1, fontname2, fontname3 etc."
```

The browser will check the user's system for the requested font in the order specified.

Color

Specifies the font color as an RGB hexadecimal number, for example,

```
COLOR=#FF0000
```

This would produce a red colored text.

Styling Attributes

Elements can also take attributes that style and format the document. The following table is a partial list of styling and formatting attributes.

HTML Attribute	Definition
BGCOLOR	Can be used on the **<BODY>** tag or any of the table tags to provide a background color.
BACKGROUND	Takes a URL as its value. Can be used on any of the table or body tags to provide a tiled image.
LINK, VLINK, ALINK	Sets the color of a link using an RGB hexadecimal.
TEXT	Used on the **<BODY>** tag. Sets the text color for the page.
LEFTMARGIN, RIGHTMARGIN and TOPMARGIN	The syntax is **LEFTMARGIN**=***n***, where ***n*** is the size of the margin expressed in pixels. Only works on IE.
CLEAR	An attribute that can be used with the ** ** or **<BODY>** tags, it takes the values **CLEAR=LEFT | RIGHT | ALL | NONE**. If, for example, the tag **<BR CLEAR=LEFT>** is employed, the browser will wait until there is a clear margin on the left before rendering the text. The pipestem symbol "**|**" indicates alternatives.
ALIGN	Applies to numerous elements and is used to align the element. The syntax is **ALIGN=LEFT | RIGHT | CENTER**

Whereas the following table refers to styling attributes associated with tables:

HTML Attribute	Definition
BORDER	Sets the border width in pixels.
BORDERCOLOR	Sets the border color.
CELLPADDING	**CELLPADDING** is the amount of space between the cell content and the border.

HTML Attribute	Definition
CELLSPACING	**CELLSPACING** is the amount of space between individual cells.
HEIGHT	Specifies the height either as a percentage or in pixels.
WIDTH	Specifies the width either as a percentage or in pixels.

Positioning

This was a perennial headache for the web author, until the advent of style sheets. Let's take a look at a few of the tricks and workarounds they came up with in the interim.

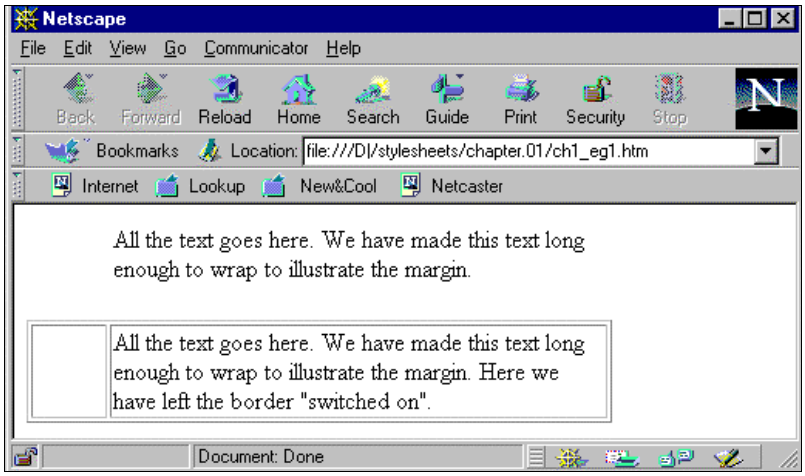
Using Tables

Because authors felt the need to control the placement of their content on the screen they took to using tables to do this. This worked fairly well but at the expense of complicating document design and making the source document difficult to read.

The following code would produce a nice margin down the left side of any written material.

```
<TABLE WIDTH=80%>
<TR>
<TD WIDTH=15%>
<BR>
</TD WIDTH=80%>
<TD>All the text goes here. We have made this text long enough to wrap to
illustrate the margin.
</TD>
</TR>
</TABLE>
```

The following screenshot shows the above in Communicator.



We have also repeated the code with the border switched on by setting:

```
<TABLE WIDTH=80% BORDER=1>
```

Note, we have added content in the form of a **
** to the empty table cell, otherwise some browsers will not render the cell.

So in order to position text accurately on the page, the web page author would have to create multiple tables and then ensure they were invisible to the viewer. Not terribly intuitive. Before the arrival of style sheets, the only alternative to this was to use the single pixel trick, which we cover next.

Using the Single Pixel Trick

Although tables give quite a bit of control over positioning, to precisely position content without using style sheets, we must resort to the single pixel trick.

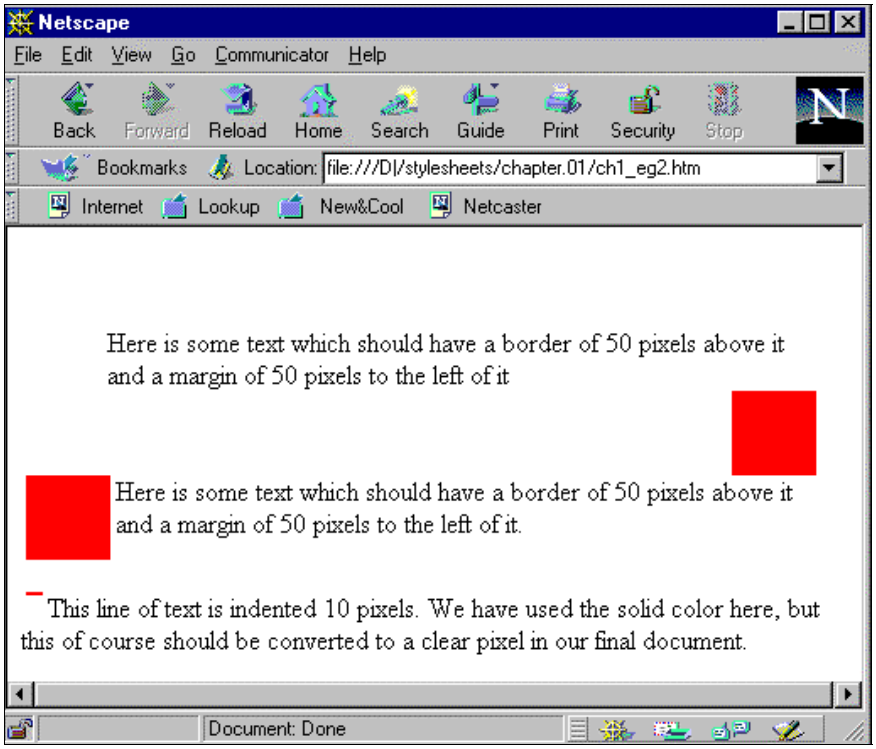
The trick here is to make a **.gif** image consisting of a single pixel, then make that pixel transparent. We then use the **VSPACE** and **HSPACE** attributes to provide a border round the single pixel.

We can use this trick for precise positioning of objects and text, and also to make sure that our tables are the precise dimensions we want them.

```
<HTML>
<BODY BGCOLOR=WHITE>
   <IMG SRC="dot_clear.gif " HSPACE=25 VSPACE=25 ALIGN=RIGHT><BR>
   <BR CLEAR=RIGHT>
   <IMG SRC="dot_clear.gif" HSPACE=25 VSPACE=25 ALIGN=LEFT>
   Here is some text which should have a border of 50 pixels above it and a
margin of 50       pixels to the left of it
   <IMG SRC="dot_red.gif" HEIGHT=50 WIDTH=50 ALIGN=RIGHT><BR>
   <BR CLEAR=RIGHT>
   <IMG SRC="dot_red.gif" HEIGHT=50 WIDTH=50 ALIGN=LEFT>
   Here is some text which should have a border of 50 pixels above it and a
margin of 50       pixels to the left of it.
   <BR CLEAR=LEFT>
   <BR>
   <IMG SRC="dot_red.gif" HEIGHT=2 WIDTH=10 ALIGN=LEFT>
   This line of text is indented 10 pixels. We have used the solid color
here, but this of     course should be converted to a clear pixel in our
final document.
</BODY>
</HTML>
```

The above code shows how to accomplish this using the single pixel trick. In the second paragraph of text we have used a red colored single pixel with height and width properties so that you can visualize the process.

In fact you are encouraged to do this while you are laying out your documents, just remember to halve the width and height when you substitute the **VSPACE** and **HSPACE** attributes, because these are added to both sides of the single pixel image.
The following screenshot shows what this looks like in practice.



Using a combination of tables and the single pixel trick it is really possible to lay out a page any way we want. The problem with this approach is that the HTML source rapidly becomes cluttered up with styling tags. To be effective each page really has to be hand crafted.

To change the look and feel of a series of pages would be a huge task. In fact the effort would be almost as great as designing the pages in first place, without the excitement that goes with creativity.

The single pixel trick is a hack, a useful hack but a hack none the less. Thank goodness style sheets will obviate the need for it.

Using Images

The final way to make sure that the page is presented just as we want it is to convert the content to an image. However, this is a very wasteful use of resources. For example, the very simple page above is 450 bytes when sent as HTML and 15,000 bytes when sent as a 16-color **`.gif`**.

Again images can be expensive (good artists don't come cheap), they take time to download plus, you can't search on them, you can't copy the text, you can't change the size of the image there are major maintenance problems and so on and so on.

Image elements are best reserved for headings with special fonts and for visual clipart, **`.gif`**, and **`.jpg`** images.

Switches and Style Blocks

It is customary to think of a tag as a switch that switches various styles on and off. This may be a valid way of thinking of style if one is writing a simple text document, but it is a poor thought process to use when we are styling anything more sophisticated.

Whether we are designing a page for a magazine, a newspaper, or the Web, we are moving blocks of text and images around our page. It is, therefore, better to think of styling in terms of styling blocks of content.

All the styling languages make use of this concept, so we should force ourselves to think in these terms straight off. Cascading Style Sheet language (CSS) uses the box concept, eXtensible Style Language (XSL) and Document Style Semantics and Specification Language (DSSSL) use the flow-object concept, but they mean the same thing—blocks of material to which we can apply styling.

Experience from teaching style languages tells me that, although this is an easy concept for beginners to grasp, it can be a difficult concept for experienced HTML buffs to adapt to. So if you fall into the latter category, it would be helpful to start thinking in terms of blocks of content rather than switches.

An Example of Styling with HTML

In order to understand what kind of styling is possible with old HTML let's look at an example. A good and fairly simple example would be a page from a Wrox Instant book. We have chosen page 82 from *Instant HTML Programmer's Reference*. Read the text on this page as well, it is applicable. The code for this page is listed as **`Ch1_1.htm`**.

First let's set the page background. The following line provides a soothing mint green.

```
<BODY BGCOLOR="#EEFFEE"><!--A pleasant mint green-->
```

Now we need to set the outline of the page. The best way to do this is via a table with a border. The width of the page should be about 6 inches. As we can only depict the width in pixels, in HTML we need a ruler and experimentation to carry this out. As a rule of thumb there are about 80 pixels to a logical inch, but the actual pixel size will depend on your monitor.

> *The "logical inch" is what your monitor thinks an inch should be. When I ask the browser to draw a box 6in. wide on my monitor the actual measured width is 8.5in, which makes the "logical inch" about 1.4 actual inches on my monitor. The measurement in the y direction may even be different. Most programming languages have a function that allows you to get this information, in VB it is* `Screen.TwipsPerPixelX` *and* `TwipsPerPixelY.` *It is likely that XSL will have a similar function built into it, but it is unlikely that we will be able to have such precise control in CSS.*

What we come up with is the following:

```
<!--The page itself-->
<TABLE BORDER="1" BGCOLOR="white">
```

```
<TR>
<TD WIDTH="500">
<BR>
```

```
<IMG SRC="dot_clear.gif" VSPACE="418" ALIGN="left">
```

The page color of course is white, the border of the table we have depicted as one pixel thick. Because we wanted to make sure of the height of the page as nine inches, we used the Single Pixel Trick, which we demonstrated earlier in this chapter.

Now we want to set the various elements in the page, and to get these set in exactly the right position we will nest in our main table some more tables and use the single pixel trick over and over again.

First we will put in the page heading. A good trick is to leave the border switched on until we have got the material exactly where we want it, then switch it off to display the completed page.

```
<TABLE BORDER="0" BGCOLOR="white">
<!--page heading type-->
<TR>
   <TD>
      <IMG SRC="dot_clear.gif" HSPACE="8" ALIGN="left">
      <FONT SIZE="2" FACE="arial">
      <B>Chapter 4 - Images and Inclusions</B>
   </TD>
</TR>
</TABLE>
```

The first lot of text is a regular paragraph. If we take care with our table construct we can use it as a template for all our other paragraphs.

```
<TABLE WIDTH=480 CELLPADDING="10" BORDER="0" BGCOLOR="white">
<!--para type-->
<TR>
   <TD>
      <FONT SIZE="2" FACE="times new roman">
      <IMG SRC="dot_clear.gif" VSPACE="8" HSPACE="5" ALIGN="left">
      In fact this page uses a table to seperate the page......etc
   </TD>
</TR>
</TABLE>
```

The **VSPACE** attribute will have to be set for each individual paragraph. Note how we have used the **CELLPADDING** attribute to provide space above and below the paragraph. We could also have used the single pixel trick, but this way is neater.

As an exercise for you, we have not made **VSPACE** high enough and so the third line does not line up with the others. Try adjusting it so that every thing looks just right.

Now to format the block of code shown on the page.

```
<!--Set the gray box in from the edge of the page.-->
<IMG SRC="dot_clear.gif"VSPACE="12" HSPACE="14" ALIGN="left">
```

```
<TABLE WIDTH=430  BORDER="0" BGCOLOR="#CCCCCC">
<TR>
   <TD>
<!--code type-->
      <FONT SIZE="1" FACE="courier new">
```

```
      <IMG SRC="dot_clear.gif"VSPACE="12" HSPACE="12" ALIGN="left">
      &lt;IMG SRC="world.gif" ALIGN="LEFT" WIDTH=75 HEIGHT=75&gt;<BR>
      &lt;B&gt;The internet is the most cost effective way to <BR>
      advertise &lt;I>your&lt;/I&gt; business today ... etc
   </TD>
</TR>
</TABLE>
```

Note the use of the escape entities for **<** and **>**. There are no new tricks here just a lot of fiddling about. The same with the heading

```
<TABLE BORDER="0" BGCOLOR="white">
<TR>
   <TD>
      <IMG SRC="dot_clear.gif" HSPACE="8" ALIGN="left">
      <FONT SIZE="3" FACE="arial">
<!--Sub Heading 2 type-->
      <I><B>Using the HTML 4.0 'float' Style Property</B></I>
   </TD>
</TR>
</TABLE>
```

One of the problems we run into is that in HTML we can't set the font size exactly, we have to use a scale of 1-7, where 3 corresponds to what the default value for the browser usually is.

Again we have to fiddle around and experiment to make things look right.

The rest of the page doesn't introduce any thing new. Here's the complete code for the page.

```
<HTML>
<TITLE> Chapter1 Tutorial1. A Wrox Page in HTML.</TITLE>
<BODY BGCOLOR="#EEFFEE"><!--A pleasant mint green-->

<!--set our page in about 1" from the left of the screen-->
<IMG SRC="dot_clear.gif" HSPACE="30" ALIGN="left">

<!--The page itself-->
<TABLE BORDER="1" BGCOLOR="white">
<TR>
<TD WIDTH="500">
<BR>
<IMG SRC="dot_clear.gif" VSPACE="418" ALIGN="left">

<TABLE BORDER="0" BGCOLOR="white">
<!--page heading type-->
<TR>
<TD>
<IMG SRC="dot_clear.gif" HSPACE="8" ALIGN="left">
<FONT SIZE="2" FACE="arial">
<B>Chapter 4 - Images and Inclusions</B>
</TD>
</TR>
</TABLE>
```

```
<TABLE WIDTH=480 CELLPADDING="10" BORDER="0" BGCOLOR="white">
<!--para type-->
<TR>
<TD>
<FONT SIZE="2" FACE="times new roman">
<IMG SRC="dot_clear.gif"VSPACE="8" HSPACE="5" ALIGN="left">
In fact this page uses a table to seperate the page into two sections.
We'll look at how tables work in chapter 6. In the meantime the important
line is the one that inserts the left-hand image:
</TD>
</TR>
</TABLE>

<!--Set the gray box in from the edge of the page.-->
<IMG SRC="dot_clear.gif"VSPACE="12" HSPACE="14" ALIGN="left">
<TABLE WIDTH=430  BORDER="0" BGCOLOR="#CCCCCC">
<TR>
<TD>
<!--code type-->
<FONT SIZE="1" FACE="courier new">
<IMG SRC="dot_clear.gif"VSPACE="12" HSPACE="12" ALIGN="left">
&lt;IMG SRC="world.gif" ALIGN="LEFT" WIDTH=75 HEIGHT=75&gt;<BR>&lt;B&gt;The
internet is the most cost effective way to <BR>advertise
&lt;I>your&lt;/I&gt; business today ... etc
</TD>
</TR>
</TABLE>

<TABLE WIDTH=480 CELLPADDING="10" BORDER="0" BGCOLOR="white">
<!--para type-->
<TR>
<TD>

<FONT SIZE="2" FACE="times new roman">
<IMG SRC="dot_clear.gif"VSPACE="40" HSPACE="8" ALIGN="left">
You can see that we've used the value <B> "LEFT" </B> for the <B>ALIGN</B>
attribute (the quotaton marks are in fact optional, because the value is a
single word with no spaces). This causes the image to be aligned with the
left margin, and the text wraps to the right of it. If it appears to be too
close, we could use the <B>HSPACE</B> and the <B>VSPACE</B> attributes to
give it more room, although in our case we wanted it to wrap as closely as
possible.
</TD>
</TR>
</TABLE>

<TABLE BORDER="0" BGCOLOR="white">
<TR>
<TD>
<IMG SRC="dot_clear.gif" HSPACE="8" ALIGN="left">
<FONT SIZE="3" FACE="arial">
<!--Sub Heading 2 type-->
```

```
<I><B>Using the HTML 4.0 'float' Style Property</B></I>
</TD>
</TR>
</TABLE>
```

```
<TABLE WIDTH=480 CELLPADDING="10" BORDER="0" BGCOLOR="white">
<TR>
<TD>
<!--para type-->
<FONT SIZE="2" FACE="times new roman">
<IMG SRC="dot_clear.gif"VSPACE="16" HSPACE="8" ALIGN="left">
Of course, we should be using the new HTML 4.0 style sheet standards to
place our image, instead of the attributes of the <B>&lt;IMG&gt;</B>
element directly:
</TD>
</TR>
</TABLE>

<!--Set the gray box in from the edge of the page.-->
<IMG SRC="dot_clear.gif"VSPACE="12" HSPACE="14" ALIGN="left">
<TABLE WIDTH=430  BORDER="0" BGCOLOR="#CCCCCC">
<TR>
<TD>
<!--code type-->
<FONT SIZE="1" FACE="courier new">
<IMG SRC="dot_clear.gif"VSPACE="12" HSPACE="12" ALIGN="left">
&lt;IMG SRC="world.gif" <BR>
<!--indent STYLE>-->
<IMG SRC="dot_clear.gif" HSPACE="12" ALIGN="left">
STYLE="float:left;width:75, height:73; &gt;"
</TD>
</TR>
</TABLE>

<TABLE WIDTH=460 CELLPADDING="10" BORDER="0" BGCOLOR="white">
<TR>
<TD>
<!--para type-->
<FONT SIZE="2" FACE="times new roman">
<IMG SRC="dot_clear.gif"VSPACE="30" HSPACE="8" ALIGN="left">
Here, the CSS1 <B>float</B> property is used to move the element to the
left margin of the page and wrap the text round it. This will only work in
browsers that support the CSS1 standard, but you could always take the
'belt and braces' approach and include both the direct attributes and the
style attribute with the matching properties:
</TD>
</TR>
</TABLE>

<!--Set the gray box in from the edge of the page.-->
<IMG SRC="dot_clear.gif"VSPACE="12" HSPACE="14" ALIGN="left">
<TABLE WIDTH=430  BORDER="0" BGCOLOR="#CCCCCC">
<TR>
<TD>
<!--code type-->
<FONT SIZE="1" FACE="courier new">
<IMG SRC="dot_clear.gif"VSPACE="12" HSPACE="12" ALIGN="left">
```

```
&lt;IMG SRC="world.gif" ALIGN="LEFT" WIDTH=75 HEIGHT=73 <BR>
<!--indent STYLE>-->
<IMG SRC="dot_clear.gif" HSPACE="12" ALIGN="left">
STYLE="float:left;width:75, height:73; &gt;
```

```
</TD>
</TR>
</TABLE>

<TABLE WIDTH=480 CELLPADDING="10" BORDER="0" BGCOLOR="white">
<TR>
<TD>
<!--para type-->
<FONT SIZE="2" FACE="times new roman">
<IMG SRC="dot_clear.gif"VSPACE="16" HSPACE="8" ALIGN="left">
 The only problem now is that behavior may be erratic in browsers that
partly support CSS1. If they don't handle it properly, and it takes
precedence over the direct properties (as it should), the results could be
less than appealing.
</TD>
</TR>
</TABLE>

<TABLE BORDER="0" BGCOLOR="white">
<TR>
<TD>
<!--Sub Heading 1 type-->
<IMG SRC="dot_clear.gif" HSPACE="8" ALIGN="left">
<FONT SIZE="4" FACE="arial">
<B>The Single Pixel GIF Trick</B>
</TD>
</TR>
</TABLE>

<TABLE WIDTH=480 CELLPADDING="10" BORDER="0" BGCOLOR="white">
<TR>
<TD>
<!--para type-->
<FONT SIZE="2" FACE="times new roman">
<IMG SRC="dot_clear.gif"VSPACE="30" HSPACE="8" ALIGN="left">
 This is a useful trick that many web authors use to achieve precise
control over layout and formatting. It's a little out of date now, because
you should be using style sheets (see the previous chapter) to control the
placement and alignment of all the elements in your pages. However, until
more browsers fully support the style sheet proposals, it can be useful.
</TD>
</TR>
</TABLE>

<TABLE WIDTH=480 CELLPADDING="10" BORDER="0" BGCOLOR="white">
<TR>
<TD>
<!--para type-->
<FONT SIZE="2" FACE="times new roman">
<IMG SRC="dot_clear.gif"VSPACE="38" HSPACE="8" ALIGN="left">
 It basically works like this. You have an image, in this case
<B><CODE>dot_clear.gif</CODE></B> which consists of one pixel. The
pixelcolor is defined as being the invisible, a trick that GIF version 89a
files can achieve. When you want to add a precise amount of space across or
down, you insert the image and use the <B>HSPACE</B> and/or the
<B>VSPACE</B> attributes to move the following elements around as required.
The code is something like this:
```

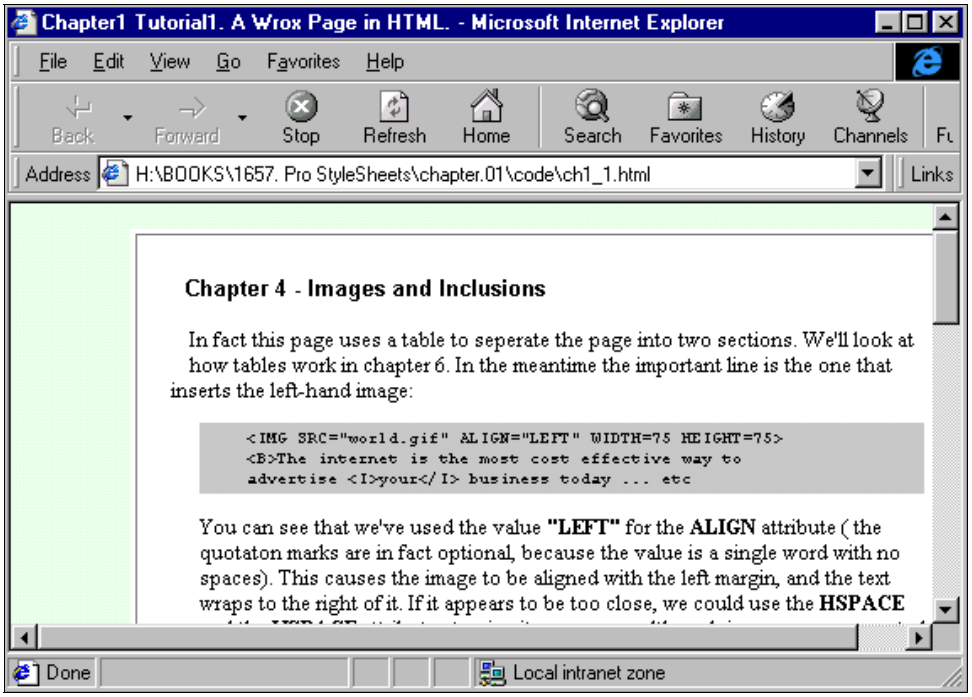
```
</TD>
</TR>
</TABLE>

<!--Set the gray box in from the edge of the page.-->
<IMG SRC="dot_clear.gif"VSPACE="1" HSPACE="14" ALIGN="left">
<TABLE WIDTH=430  BORDER="0" BGCOLOR="#CCCCCC">
<TR>
<TD>
<!--code type-->
<FONT SIZE="1" FACE="courier new">
<IMG SRC="dot_clear.gif"VSPACE="12" HSPACE="12" ALIGN="left">
&lt;IMG SRC="dot_clear.gif" HSPACE=x VSPACE=y &gt;
</TD>
</TR>
</TABLE>

<!--PAGENUMBER.-->
<IMG SRC="dot_clear.gif"VSPACE="32" HSPACE="2" ALIGN="left">
<TABLE WIDTH=50  BORDER="0" BGCOLOR="white" >
<TR>
<TD VALIGN=center>

<FONT SIZE="4" FACE="arial">
<IMG SRC="dot_clear.gif"VSPACE="18"  >
82<BR>
</TD>
</TR>
</TABLE>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

The following screenshot shows how it displays on IE4.



It looks the same on Communicator. On my monitor it is 6 inches wide, it may be more or less on your monitor.

So, styling with HTML is complex, messy and time consuming. To see a summary of problems facing the web designer before the advent of style sheets, check out an article called *"Severe Tire Damage on the Information Superhighway"* by David Siegel, which you'll find at: **`http://www.dsiegel.com/damage/index.html`**.

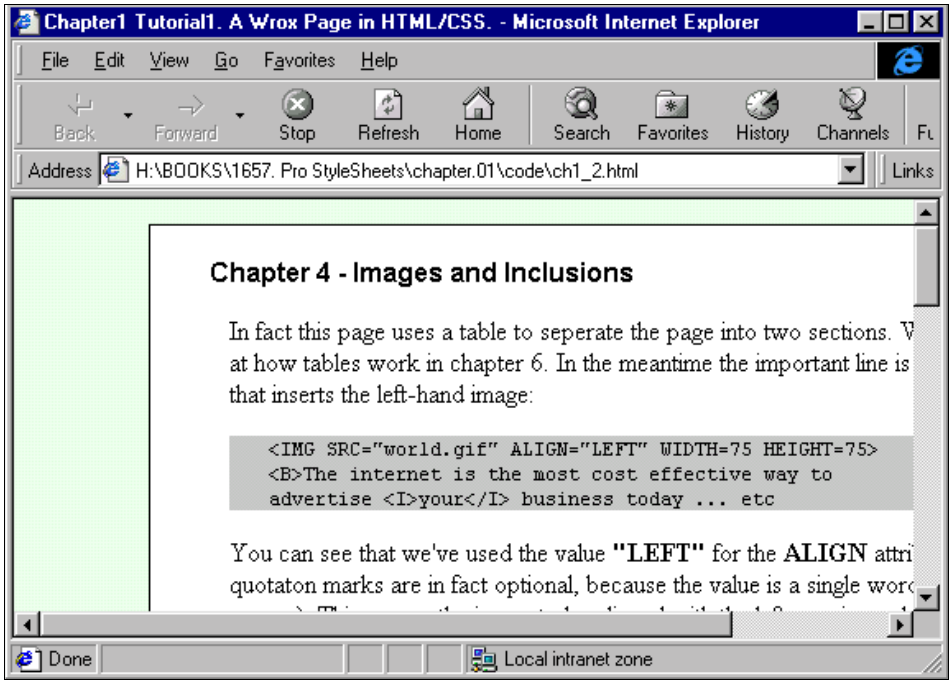
OK, we've seen the problems, now let's see the solution.

Styling with Style Sheets

Style sheets changed all this. They made styling much quicker, easier and cleaner. We aren't going to cover them in detail in this chapter, we'll leave that until Chapter 2. We are just going to show you how great they are by repeating the same tutorial using a style sheet instead of HTML.

An Example of Styling with Style Sheets

The picture below shows how our page that has been styled using HTML and a style sheet looks.



The next figure shows the HTML.

```
<HTML>
<HEAD>
<TITLE> Chapter1 Tutorial1. A Wrox Page in HTML/CSS.</TITLE>
<LINK REL="stylesheet" TYPE="text/css" HREF="ch1_2.css">
</HEAD>
<BODY>
<DIV CLASS="page">

<DIV CLASS="phead">
Chapter 4 - Images and Inclusions
</DIV>

<DIV CLASS="para">
In fact this page uses a table to seperate the page into two sections.
We'll look at how tables work in chapter 6. In the meantime the important
line is the one that inserts the left-hand image:
</DIV>

<DIV CLASS="code">
&lt;IMG SRC="world.gif" ALIGN="LEFT" WIDTH=75 HEIGHT=75&gt;<BR>
&lt;B&gt;The internet is the most cost effective way to <BR>
advertise &lt;I>your&lt;/I&gt; business today ... etc
</DIV>

<DIV CLASS="para">
```

```
You can see that we've used the value <B> "LEFT" </B> for the <B>ALIGN</B>
attribute (the quotation marks are in fact optional, because the value is a
single word with no spaces). This causes the image to be aligned with the
left margin, and the text wraps to the right of it. If it appears to be too
close, we could use the <B>HSPACE</B> and the <B>VSPACE</B> attributes to
give it more room, although in our case we wanted it to wrap as closely as
possible.
</DIV>

<DIV CLASS="head2">
Using the HTML 4.0 'float' Style Property
</DIV>

<DIV CLASS="para">
Of course, we should be using the new HTML 4.0 style sheet standards to
place our image, instead of the attributes of the <B>&lt;IMG&gt;</B>
element directly:
</DIV>

<DIV CLASS="code">
&lt;IMG SRC="world.gif" <BR>
<!--indent STYLE>-->
<SPAN CLASS="indent">
STYLE="float:left;width:75, height:73; &gt;</SPAN>
</DIV>

<DIV CLASS="para">
Here, the CSS1 <B>float</B> property is used to move the element to the
left margin of the page and wrap the text round it. This will only work in
browsers that support the CSS1 standard, but you could always take the
'belt and braces' approach and include both the direct attributes and the
style attribute with the matching properties:
</DIV>

<DIV CLASS="code">
&lt;IMG SRC="world.gif" ALIGN="LEFT" WIDTH=75 HEIGHT=73 <BR>
<!--indent STYLE>-->
<SPAN CLASS="indent">
STYLE="float:left;width:75, height:73; &gt;</SPAN>
</DIV>

<DIV CLASS="para">
The only problem now is that behavior may be erratic in browsers that
partly support CSS1. If they don't handle it properly, and it takes
precedence over the direct properties (as it should), the results could be
less than appealing.
</DIV>

<DIV CLASS="head1">
<B>The Single Pixel GIF Trick</B>
</DIV>

<DIV CLASS="para">
This is a useful trick that many web authors use to achieve precise control
over layout and formatting. It's a little out of date now, because you
should be using style sheets (see the previous chapter) to control the
```

```
placement and alignment of all the elements in your pages. However, until
more browsers fully support the style sheet proposals, it can be useful.
</DIV>

<DIV CLASS="para">
It basically works like this. You have an image, in this case
<B>dot_clear.gif</B> which consists of one pixel. The pixelcolor is defined
as being the invisible, a trick that GIF version 89a files can achieve.
When you want to add a precise amount of space across or down, you insert
the image and use the <B>HSPACE</B> and/or the <B>VSPACE</B> attributes to
move the following elements around as required. The code is something like
this:
</DIV>

<DIV CLASS="code">
&lt;IMG SRC="dot_clear.gif" HSPACE=x VSPACE=y &gt;
</DIV>

<DIV CLASS="pnum">
82
</DIV>
<BR>
<!--end of div class=page-->
</DIV>
</BODY>
</HTML>
```

Finally there is the style sheet.

```
/* style sheet for ch1_2.htm*/


BODY{

background-color:#EEFFEE;
}

DIV.page{


background-color:white;
margin-left:0.75in;
border:solid 1px;
width:6in;
}

DIV.phead{
font:bold 12pt arial,sans-serif;
margin-top:0.2in;
margin-left:0.375in;

background-color:white;
}
```

```
DIV.para{
font:normal 12pt 'times new roman',serif;
margin-left:0.5in;
```

```
margin-top:1em;
margin-bottom:1em;
margin-right:0.5in;
background-color:white;
}

DIV.code{
font:normal 9pt 'courier new',monospaced;
margin-left:0.5in;
padding-left: 0.25in;
margin-right:0.5in;
border:none 1px;
background-color:#CCCCCC;
}

DIV.head1{
font:bold 16pt arial,sans-serif;
margin-left:0.375in;
background-color:white;
}

DIV.head2{
font:bold italic 12pt arial,sans-serif;
margin-left:0.375in;
background-color:white;
}

DIV.pnum{
font:bold 16pt arial,sans-serif;
margin-top:0.25in;
padding-bottom:0.25in;
margin-left:0.25in;

}

SPAN.indent{
margin-left:0.4in;
}
```

The output pages are virtually identical. The page using a style sheet looks better because it is 6 virtual inches across (8.5 real inches on my computer screen) as opposed to 6 actual inches in the page styled with HTML tags.

Look at the HTML page and compare it with the tagged page. This page is actually readable without much effort.

Now look at the Style Sheet page and, although you may not know any CSS, at this stage you can tell at a glance that the styling language is logical, understandable and precise. We'll go into detail on how to create and apply style sheets in the next chapter.

HTML Styling versus Style Sheet Styling

First, let's examine the problems that come out of our HTML styling example.

Time is Money

In the first example above we tackled a reasonably straight forward job of formatting a file using some tried and trusty HTML guru work-arounds and tricks, and the result is just as we wanted it.

The problem is connected to how long the entire exercise took. I started off by just typing in the text, (I'm a slow typist, so I didn't want that to flavor this experiment) and then timed how long it took to get every thing right. It took almost two hours. I then tried another page using the templates that I had already created, and it took me almost an hour. Calculate that for the book. 400 pages times 3/4 hour (assume I get better as I go along), times $25 an hour (and that's cheap). The end result is $7500 to put up a web site containing these pages in this format.

> *Actually most commercial pages, are generated on-the-fly using script, or are generated using proprietary software.*

Also look and see how difficult it is to read the actual content in all that formatting, consider that intranets are putting up hundreds of pages of information a day, and you realize that we have a major problem with HTML type styling. The answer of course is style sheets. Later on we will consider the advantages of CSS type style sheets, but before doing this lets look at some of the other problems associated with the current HTML type of styling.

Download Speeds

Most pages use a lot of images to compensate for the difficulty of styling with HTML tags, and this takes time to download. As an experiment try going to a graphics heavy site, and navigate between their pages. For example:

`http://www.microsoft.com`

On my 56 kbps modem it takes between 30 and 40 seconds for each page of this site, which makes heavy use of headers in the form of graphics, to download.

Now go to a site that uses style sheets.

`http://www.hypermedic.com/style`

After emptying my cache, it took less than a second for each page to download, making for very fast navigation.

Admittedly some sites need to be graphics heavy, but users certainly appreciate the speed that style sheets bring.

Even if we don't use graphics, but use HTML type styling, the pages themselves are larger. The Wrox page in the example above is 7K. This is twice the size of the page using style sheets and those extra bytes all come from the styling tags.

Admittedly the style sheet file that is necessary to display non HTML styled pages weighs in at 1K, but this same sheet could be used for all 400 pages of the book. This represents an overall saving of about 1.5 megabytes for a single simple documentation project.

Maintenance

It is great fun to design and build pages, but unfortunately in the real world they have to be maintained, and maintaining pages with a lot of HTML styling can be a nightmare. The following list charts the various parts of said nightmare.

- First the content of the pages can be difficult to read because of all the styling elements and tags.
- Secondly if we change the content, the balance of the page is often upset, and we have to alter our tags and attributes and get things right by trial and error.
- Thirdly there is no easy way to alter the overall style of a set of pages without altering every tag individually, usually by hand.
- In fact it is often easier to write a one-off program to handle the problem with code or script.

Style sheets have none of these disadvantages, and have the additional advantages set out below.

The single style sheet we developed above can be applied to as many documents as you want with a single line of code in the HTML document—`<LINK REL="stylesheet" TYPE="text/css" HREF="ch1_2.css">`.

- If you wish to change any of the styling in a family of documents, you just have to alter one document, the style sheet, not thousands of HTML sources.
- The size of your HTML document is almost half of the document written up in tagged HTML.
- It makes it possible to separate authoring, and styling. Your copywriters can concentrate on the copy, and your artists/typographers can concentrate on the styling.
- You will save both time and money.
- If you really want extra kudos, suggest to your boss what styling all your pages in XML/XSL will do for productivity, profitability, and his stock options. But I am afraid you will have to read further before you can pitch this line to him/her.

Why did Styling get so Complicated in HTML?

To answer this question we have to go back to the origins and original purposes of HTML.

HTML was originally designed for the transfer of referenced (with links) scientific documents across numerous different platforms and the Internet. It is difficult to realize now that in the very recent past Windows was not so dominant a platform.

The original HTML was very good at what it was designed to do, namely transfer document content. However, somewhere along the line it crossed a divide from being a mere content provider for scientists to being a means of communication for the average consumer. Most are agreed this process started with the first Mosaic browser which allowed transfer of images, and accelerated with the release of Netscape 1.1.

The "Old Guard" fought an unsuccessful rearguard action to maintain the so-called "purity" of the Web, but as soon as commercial forces became involved, they were doomed to failure. They were done in by self-proclaimed "web-terrorists" such as David Siegel who were intent on making the Web a stylish and consumer oriented place. The web-terrorists were provided with tools by

"gun-runners" such as Marc Andreessen intent on making the Netscape browser the dominant browser. His chief tool to accomplish this was to create proprietary tags to make the Web a more stylish and attractive place.

Once Microsoft realized that they had almost missed the boat, they threw their massive resources into the struggle, again using proprietary styling and Multimedia tags as their chief weapon. The purist's battle to keep the Web content driven was doomed—and styling tag followed styling tag. Individuals such as David Siegel invented new tricks like the single pixel gif, new uses were found for tables, layout and styling improved, but all at the expense of document clarity.

While David Siegel took to boasting "the Web is ruined and I ruined it", the W3 consortium lagged behind the browsers to such an extent that the HTML 3 proposals were never finished, becoming out of date before they ever became a recommendation! And so the web and HTML became a disordered mishmash of styling tags and guru hacks. The next chapter shows how this dilemma was resolved using Cascading Style Sheets.

Summary

In this chapter we looked at various forms of web page styling.

- We saw how styling markup differs from semantic and structural mark up, and how using mark up to style is different from using a style sheet.
- We looked briefly at some of the more common styling tags and tricks used in HTML, and then used them to style a relatively simple document.
- We looked at some of the problems that style sheets should be asked to solve, and looked at how CSS in particular offers a solution.
- We enumerated the advantages of using style sheets over styling markup.

In the next chapter we will jump right into basic CSS style sheets, and then in the rest of the first part of this book we will have a look at basic XML and how CSS can be used with it. We will also take a first look at XSL, a style language that is very much in the formative stages, but will probably be used to add powerful styling capabilities to XML.

Exercises

1. Take any 'commercial' page on the Web, and remove all the images and formatting tags. See what you are left with.
2. Try and make the page interesting again, just using text.
3. Take a series of pages on a site, and time the average download time on your site.
4. Now time the average amount of time you spend reading the pages. (Research suggests that we spend 70% of our time waiting for pages to download, 30% of our time reading.)