by Peter Collinson, **Hillside Systems**

DENISE ORTAKALES

Tables on the Web

Since the earliest days of HTML, there has been what might be described as “a battle of approaches” between the folks that generate the HTML standards and the users of those standards. By users, I mean both Web page authors and the browser vendors.

The people who create the standards see HTML as a markup language, a way of specifying relatively high-level information about the text. HTML uses `<P>` to mean “new paragraph,” and it’s up to the browser to use that information to implement the “look” of a paragraph. The view is that it should always be the responsibility of the program that displays the HTML to provide the actual formatting, making its own decisions about the placement and size of text or images based on the high-level markup information. So, for a paragraph, the browser could have a selection box allowing the user to specify whether they want to see paragraphs as left-justified blocks of text or whether they want the first line of a paragraph to be indented a little.

The aim of the standards body, the World Wide Web Consortium (W3C), has been to provide more flexibility in formatting, not by implementing new formatting tags but by separating the content markup, the HTML, from the formatting information that is placed in a style sheet. Browsers are appearing on the market that implement Cascading Style Sheets, enabling the browser, the page author and the viewer to change the meaning of the tags by specifying how the tags should affect the content.

Style sheets will become considerably more important as the new generation of browsers come into more widespread use, but I suspect that it will be some time before page authors can confidently use them in preference to the current widespread set of tricks and hacks that are used to make Web pages look “nice.”

Actually, most page authors are not terribly interested in the theoretical notions of page markup. They just want to make their pages work on the Web, and by “work” I mean “what the author sees, the viewer gets.” Ideally, the pages should look the same to all viewers irrespective of the browser or system they are using. Many page designers would also like to be able to place objects at fixed positions on a page, so they can control the completed page design precisely. However, Web viewers all have different-size screens with different resolutions, and so being able to absolutely position objects may not be quite what is needed. As a compromise, it would be reasonable to expect that the relationships between the objects on the page should remain constant across all browsers and platforms. This is certainly far from the reality today.

So we have a clash of ideas: On one side, the standards body wants to push HTML more in the direction of carrying content and not page formatting; on the other, we have the page designers, who want to be able to have precise control of where objects are placed on the page.

Where are the browser vendors sitting in this difference of opinion? Early on, Netscape Communications Corp. realized that there were a great many deficiencies in HTML, and it successively released browsers with a growing list of nonstandard experimental features that are commonly called “Netscape extensions.” Some of these features added new markup capabilities, such as the ability to describe tables. Some added new tags that are aimed at changing the format of the HTML page. For example, you can change the size and color of the font of some random portion of the text.

Microsoft Corp. joined the fray with its Internet Explorer browser and, in a relatively short time, has managed to generate more new features than Netscape ever did. Any good book on HTML now has to spend considerable space describing the features that work in one browser and not another.

I am now walking along a narrow knife edge between my page "working" for your browser and my page looking disgusting.

What of the viewers? As each new Web browser has been released, I believe we have seen a reduction in ability of the viewer to change the look of the pages they are seeing. The “power” has shifted to the page designer. My guess is that as a viewer you have your own opinions of exactly what power you would like to have. Putting my viewer hat on, I certainly want to control the size of the browser window on my screen. I also want to control the size of the fonts that are used to render characters inside the browser window. I like to be able to change the size of the characters because I have screens of differing resolutions and want to set things up so I can see as much of the page as possible and still be able to read the text.

Early browsers gave me complete control over the sizes of the fonts that were used to display the content. I could independently alter the style and point sizes of the two available fonts: variable- and fixed-width. I note that Internet Explorer has actually reduced this control. I can now set the size from a range of T-shirt sizes: extra large, large, medium, small and extra small. I cannot select the actual point size of the fonts.

This erosion of the viewer control is good for the page designer; it makes it harder for the viewer to set up their browser so the page will not render as the designer created it. I dislike the loss of my ability to set point sizes. I find the normal setting for the point size of my chosen fixed-width font (Courier) in Internet Explorer is too small relative to the size of my selected variable-width font (Arial), whatever overall T-shirt size setting I choose. I’d like to be able to increase the Courier default size by a point or so.

Table Basics

As a page designer, I am now walking along a narrow knife edge between my page “working” for your browser and my page looking disgusting, or worse, not appearing at all. My own rule is that I try hard to avoid features that are only supported by one browser, unless I can easily detect the difference. I continue to make judgments about what “new feature” is acceptable. I suppose that I am somewhat conservative, believing that people don’t often replace programs that are apparently working, so there’s a considerable proportion of old browsers in use.

I’ve felt that it’s been safe to use HTML tables for some time now, although I don’t employ some of the newer features because I am unsure whether the browsers that implement them are in mass use yet. Tables are part of the HTML 3.2 recommended specification (current at the time I write this), which describes “the widely used subset” of the various browser implementations of tables. I talked a little about using tables to control page layout in my article, “Designing Web Pages” (December 1996, Page 28). I’ll revisit that topic in this column. But first, some basics.

The HTML table model is easy to understand. A table is a collection of *cells* arranged in rows and columns. You start a table by entering the `<TABLE>` tag and end the table with the `</TABLE>` tag. The end tag is important because browsers can deal with nested tables and need to be able to tell where tables start and end. Leaving off the end tag can sometimes lead to parts of the page mysteriously disappearing when it’s viewed.

Each table row is started with the `<TR>` tag and ended with `</TR>`. Within the row are a number of cells each started with `<TD>` and terminated with `</TD>`. You’ll find that browsers will be lax about the need to include these end tags, and your table may still be output if you omit them. However, forgetting to include end tags can cause huge problems when tables are nested, so my advice is always to use them when creating tables.

Here’s a simple, six-cell table made up of three columns and two rows:

```
<TABLE>
<TR><!-- start of row 1 -->
<TD>1</TD>
<TD>2</TD>
<TD>3</TD>
</TR><!-- end of row 1 -->
<TR><!-- start of row 2 -->
<TD>4</TD>
<TD>five</TD>
<TD>6</TD>
</TR><!-- end of row 2 -->
</TABLE>
```

Incidentally, a comment in HTML is any text enclosed between `<!--` and `-->`. The table will display as:

```
1 2    3
4 five 6
```

It will be left justified on the page, and each column and row will expand to fit the data in the cell. You can add a caption to the table by placing text inside `<CAPTION>`...`</CAPTION>` and inserting the result where a row specification would be placed. Special column and row titles can be created by using `<TH>`...`</TH>`, which specifies a *header* text cell rather than a normal `<TD>`...`</TD>` data cell. I’ve never used any of these features.

If you want the table to be centered on the page, then you must enclose it within a <CENTER>...</CENTER> pair, or place it in a paragraph that is defined with the center attribute:

```
<P ALIGN="center">
<TABLE>
<!-- table contents -->
</TABLE>
```

You can also use this method to place the table on the right-hand side of the page.

These alignment methods leave the table standing on the page with white space around it. If you want the text to wrap around the table, then you can add an ALIGN attribute.

```
<TABLE ALIGN="left">
```

will place the table on the left-hand side of the page with the text flowing round it on the right. You can replace "left" with "right" to achieve the reverse effect. Incidentally, the ALIGN attribute for TABLE is relatively new (it's part of the 3.2 specification) and may not work for all browsers.

Cell Spacing and Borders

By default, the browsers place two pixels of *cell spacing* around each cell in your table. You can control the space using the CELLSPACING attribute. Typically, when you are using a table for layout, you want to eliminate the extra space, and you can do that by using a value of zero:

```
<TABLE CELLSPACING=0>
```

The browsers also supply two more pixels as *cell padding*. The pixels are added around the data in each cell. Again, you can control the padding with an attribute. The following will turn off both spacing and padding:

```
<TABLE CELLSPACING=0 CELLPADDING=0>
```

The final attribute of interest, <TABLE BORDER>, controls whether or not a border is drawn around your table. By default, a table is displayed with no borders. You need to explicitly turn them on if you want the browser to draw lines around your table.

The browser will add a "raised" section around each cell. Browsers attempt to make the borders look three-dimensional by adding highlights and shadows. The unqualified BORDER attribute will add a single-pixel border, and you'll find that some browsers will add a single-pixel shadow inside the data cell to give it a 3D look. You can extend the BORDER attribute by giving it a value corresponding to the number of pixels of highlight that are to be added:

```
<TABLE BORDER=2>
```

The highlight and shadow decoration is added around the cell spacing in pixels, so in the above example, you will get two pixels of spacing between the highlights. A table that starts

```
<TABLE BORDER=1 CELLSPACING=0>
```

will result in a minimum-size border around your table.

Using the Width Attribute

I mostly use tables to control the layout of my pages, and this is possible because you can supply a WIDTH specification to the TABLE tag. The simplest use of the WIDTH attribute is to spread a table across a browser window, irrespective of the size the user has chosen for the window:

```
<TABLE WIDTH="100%"
CELLPADDING=0 CELLSPACING=0>
```

Now the table will occupy all the horizontal space inside a browser window that is available to it and will stretch and contract as the window is resized. If the table itself is within another table cell, it will occupy the full horizontal space that is allotted to it and, again, will size itself to fit the available space. I'll often use a table with WIDTH set to 100% when presenting a simple index page consisting of a table with links that take you into a Web site.

You can also add WIDTH specifications to columns within a table, which removes the equal spacing that is the default in most browsers. So, a two-column table with a constant relative column width can be specified as

```
<TABLE WIDTH="100%"
CELLPADDING=0 CELLSPACING=0>
<TR>
<TD WIDTH="30%">...</TD>
<TD WIDTH="70%">...</TD>
</TR></TABLE>
```

In addition to using percentages of the available screen width, you can supply absolute pixel counts for the table. This is a useful device if you want to place a table over a background with a colored left margin:

```
<TABLE WIDTH="100%"
CELLPADDING=0 CELLSPACING=0>
<TR>
<TD WIDTH=50><BR></TD>
<TD>...</TD>
</TR></TABLE>
```

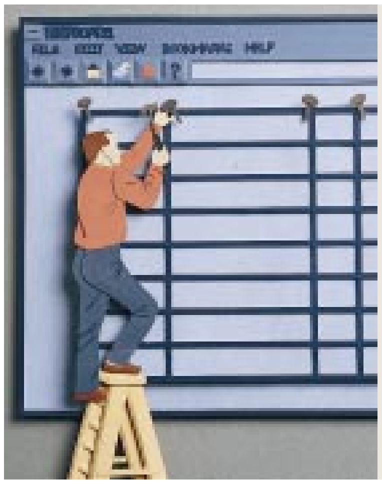
The table will stretch as the screen is resized, but the left column will remain the same absolute width of 50 pixels. To create an empty column, I tend to insert a <BR> break tag. Some people just use &nbsp;, adding a nonbreaking space into the column. Browsers tend to suppress completely empty columns.

For most of my Web pages, I like to create an empty left margin that is sometimes filled with a colored image tiled onto the background. The main contents of the page are placed in a width-constricted section on the right-hand side of the browser window. The complete page is contained in a single two-column table providing the layout. Constraining the width of the page makes any text in the page much more readable and follows traditional typesetting practice that understands that humans read short lines faster and more comfortably than long ones. I discussed these issues in my column, "Automatic Web Page Creation" (February 1997, Page 26).

My basic page template looks like this:

```
<TABLE WIDTH=450
CELLPADDING=0 CELLSPACING=0>
<TR>
<TD WIDTH=50><BR></TD>
<TD WIDTH=400>
<!-- page contents here -->
</TD>
</TR></TABLE>
```

The page is thus constrained to 450 pixels, with a 50-pixel left margin and 400 pixels occupied by the contents. I found early on that you do need to specify both width values in the `<TD>...</TD>` tags for some browsers to operate correctly.



By trying to be helpful, Internet Explorer has created a new problem for the designer.

The total screen width is somewhat arbitrary. You may wish to choose a wider value if you believe that most of your viewers use PCs. The main problem with this approach is that you not only constrain the page to a maximum width, which is a good thing, but you are also setting a minimum width, which is a bad thing. The page begins to stop being usable if the user increases the point size of the text to some large size on a small screen. Also, if the user resizes the page so that it is narrower than the total table width, then most browsers will chop off the right-hand side of the page. The browsers cannot resize the displayed area, because you've told them not to.

The problem of chopping off the right-hand side of the page led to a complaint about the USENIX site that I redesigned. First, one user complained that he couldn't get two nonoverlapped Netscape windows on the screen of his Sun and still read the full information. Second, another user complained about the general theory behind constraining the page size to an absolute pixel width. He said that he used a large font and the pages didn't work at all. I believe that what he reported was true, although, I suspected he overdramatized the problem somewhat. I felt that both complaints were justified, but that both people must be having problems with the many Web sites that are designed for PCs showing at least 600 pixels across the screen. To be honest, after I redid the USENIX site, I expected a barrage of complaints from the very vocal and outspoken USENIX community, and hearing but two, it seems that constraining the page size is an acceptable compromise needed to achieve the enhanced readability of the contents of the site.

There are, of course, other problems. One problem begins to emerge when you attempt to display data that is naturally wider than the width of the table column containing the active page content. For example, you specify a column width of 400 pixels and attempt to display an image wider than 400 pixels or, perhaps, place a fixed-width CODE section that occupies more than 400 pixels. You've presented the browser with a formatting problem. It needs to find some way of squeezing the information onto the screen.

Netscape Navigator deals with the problem by clipping the content that it wants to display using a rectangle whose width is the size of the area you have allocated. The clipping can lose information. For example, I noticed that some of an important PDP signature had been quietly clipped from a displayed page.

Internet Explorer tries to be helpful. It seems to notice that there is nothing in the left margin and will migrate the bulging page contents leftwards across the screen, creating space for the image or text that will not fit. The migration can be unwanted, especially when the active content starts being displayed over a colored image in the background that normally appears in the left margin. By trying to be helpful, Internet Explorer has created a new problem for the designer. At some point sizes, a carefully laid out page will lurch to the left, disappearing into the background. Of course, all this goes away if frames are used to create margins, but that's another can of worms, and possibly another article.

Cell Attributes

When text is written into a cell, it is formatted as if it were appearing in a browser window. Text and images will be wrapped, and all the standard formatting tags will be obeyed. As we've seen, the browsers differ in what they do when the content is too wide to fit the cell.

The inverse case is easier. If there is more space in the cell than is needed to display the contents, then by default, the text and images will be left justified and vertically centered in the cell. Attributes in the `<TD>` tag can be used to change these defaults for a single cell. The same attributes can be supplied to the `<TR>` tag, changing the attributes for the entire row.

The `ALIGN` attribute is used to change the horizontal placement of the HTML within a cell. Don't confuse this `ALIGN` with the one that may be placed within the `TABLE` tag. The `ALIGN` attribute within the `<TD>` or `<TR>` tags takes values of `left`, `right` or `center` and will perform left or right justification, or centering of the data within the cell, respectively.

The VALIGN attribute affects the vertical placement of the data within a cell. Its values are top, bottom or baseline. I make extensive use of the top attribute, ensuring that the top of the data in a cell is placed at the top of the area being rendered. The bottom value for VALIGN ensures that the data is placed at the bottom of the column. The baseline value aligns the top line of the data with the baseline of the text in other cells across the table, and is useful if your table contains different font sizes.

Apart from these controls, the data displayed in cells can sometimes be hard to format "correctly." For example, there is no easy way to line up data in a table, and it's difficult to line numbers up on decimal points, unless you use a fixed-width font, which can sometimes look ugly.

Another pair of attributes that are useful to achieve special effects are COLSPAN and ROWSPAN. Saying something like

```
<TD COLSPAN=2>
```

makes that cell span two columns across the table. The ROWSPAN attribute makes the cell span some number of rows down the table.

Finally

My current HTML bible is the Second Edition of *HTML, The Definitive Guide* by Chuck Musciano and Bill Kennedy, published by O'Reilly & Associates Inc. You can find more of the gory details on tables in this excellent book. The USENIX Web site is http://www.usenix.org. The HTML 3.2 specification, a W3C recommendation, is available on the Web at http://www.w3.org/TR/REC-html32.html.

***Peter Collinson** runs his own UNIX consultancy, dedicated to earning enough money to allow him to pursue his own interests: doing whatever, whenever, wherever... He writes, teaches, consults and programs using Solaris running on a SPARCstation 2. Email:* pc@cpg.com.