Table manners

Forms and tables add versatility to your web page. Nik Rawlinson shows how.

Cast your mind back to our March issue, to the first part of this workshop, in which we constructed the bare bones of our site; those frames which now host the rapidly growing collection of pages that we have seen fit to inflict upon the browsing public. The structure has been maintained throughout, giving our pages a unified look and feel even though we may have used a variety of styles.

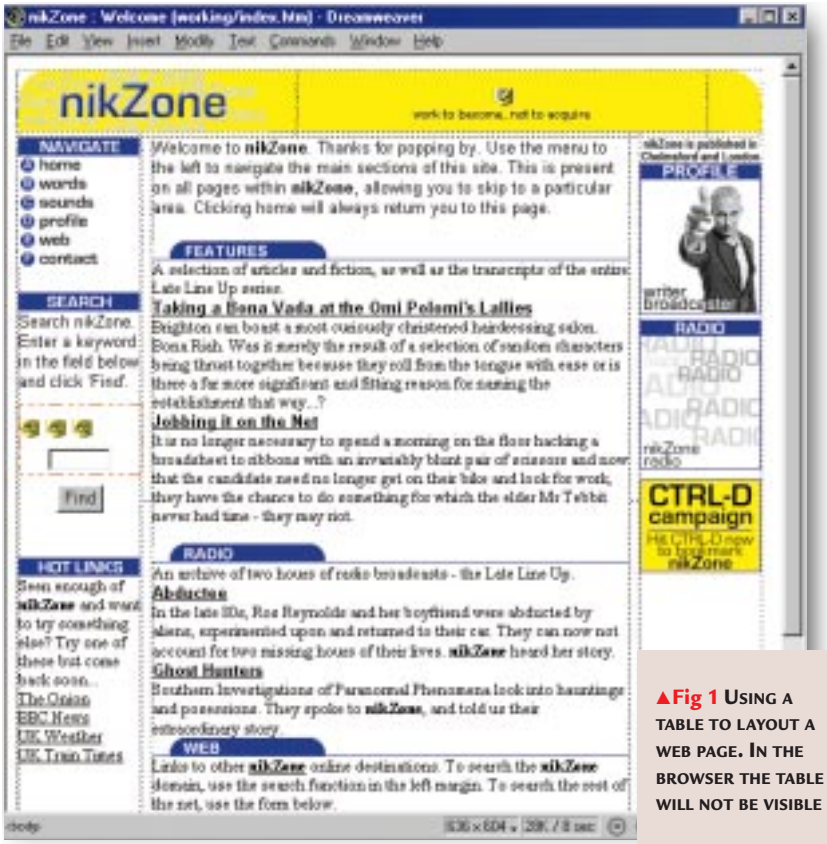
This gives us the freedom to express ourselves in as many different ways as we like without having to worry that the pages we create may have no common theme. This is because the theme is contained within the unchanging menu and header frames. It also dramatically reduces the amount of time we have to spend working on our pages as most of the work need be done only once, referred to in the initial file (usually index.htm). The second workshop in this series took a look at the most basic page elements: text, images and hyperlinks.

In this last part of the series, we'll look at tables and forms. Although neither are a necessary component of a successful site they do provide some useful facilities.

Tables are purely and simply a layout tool. Many professional authoring packages allow the user to drag and drop data from applications such as Excel or 1-2-3, dropping it onto the page as a table. But tables are far more versatile than this: they can incorporate frames, the borders of each cell can be set to a specified width, individual cells can be merged horizontally or vertically and background colours can be specified to match a corporate colour scheme.

Tables and forms provide a number of useful facilities

Let's take the fairly extreme example of table use in Fig1. I've loaded the index.htm page from my personal site at www.nikzone.com into Macromedia Dreamweaver 2. Feel free to visit the site to see how the rest of it works and the fact that it looks as though there are no tables on the page.



▲Fig 1 USING A TABLE TO LAYOUT A WEB PAGE. IN THE BROWSER THE TABLE WILL NOT BE VISIBLE

Now look at Fig 1. As you'll see, the whole page is laid out within a very extensive table with a variety of cell sizes, and many cells merge allowing the contents to spill across from one cell to the next.

The reason that the cell boundaries are visible within Dreamweaver yet invisible in the browser is that they have been set to a width of zero. Although it may not be immediately evident, this function presents us with a useful facility, namely image slicing, enabling variable compression across a large graphic.

Many graphics packages can slice graphics in a way that will separate areas of consistent colour from details. This allows the solid colours to be more tightly compressed as they will not suffer quite so badly from loss of quality. Lower compression, meanwhile, will be applied to detailed areas to maintain a level of quality. A table with borders set to a width of zero will then be used to position the sliced sections next to each other to resemble the original image. The net benefit? Significantly faster download times because of an increased average level of compression that it would have been unwise to use on an 'unsliced' image.

Let's create our first table

We'll use it to act as a menu for the pages that would eventually form the contents of the 'words' section of the site we have been building at http://i.am/pcw/.

As it will consist of a list of pages on the left and a brief description of the content of each page on the right, we shall make a two-column table with a new row for each corresponding page. It's a small site and thankfully it is only two rows, which makes the code somewhat easier to examine. See Fig 2.

As you will see, the table is enclosed by <table> and </table> tags. With many

[FIG2]

Making a two-column table

```
<table width="400" border="0" cellspacing="0"
cellpadding="0">
  <tr>
    <td align="left"
valign="top"><font
face="Arial, Helvetica, sans-
serif" size="-1"><a
href="polari.htm">Polari</a><
/font></td>
    <td>Polari is an all but forgotten language, but in the
fifties and sixties it was used extensively and even made
regular appearances on radio.</td>
  </tr>
  <tr>
    <td align="left"
valign="top"
width="100"><font
face="Arial, Helvetica, sans-
serif" size="-1"><a
href="jobbing.htm">Jobbing</a
></font></td>
    <td>Looking for a job? Tried all the usual avenues? But
have you tried the net? More and more companies are
advertising their jobs online. We find out more...</td>
  </tr>
</table>
```

tags, positioning the closing tag can result in nothing more serious than an extended stretch of formatting; extra bold or underlined words, for example. Incorrect positioning of the closing </table> tag, however, can see your table not just extended but also falling in some rather strange locations on your page, most often at the very top or very foot of the page.

In this example [Fig 2] the <table> tags also tell the browser that the table is to be 400 pixels wide and have invisible borders (border=0) that do not interfere with the placement of the text (cellpadding =”0” cellspacing=”0”).

Each row is enclosed within <tr> and </tr>. Once again, it is important to remember to close each row at the end to avoid confusion. Not closing it will cause the browser to append the first cell of the next row to the previous row, extending that row by one cell and creating a ragged right-hand margin where not all rows conclude at the same point.

The cells themselves, in each row, are defined using the <td> and </td> tags. These are very versatile tags which can contain a number of parameters to define the appearance and behaviour of each cell. For example <td colspan=”2”> would, not surprisingly, tell the browser to make this cell two columns wide, which is an easy way of appending a title bar or section break to your table. It can also contain a colour definition. For instance, <td bgcolor=”#0000ff”> will generate a dark blue cell using the hexadecimal code for dark blue *(see the March issue workshop for a better explanation of colours)*. This is also the place to detail the alignment of the cell contents. The default positioning for cell contents is to be aligned to the left and positioned exactly half way between the top and bottom borders.

In Fig 2 we have set the vertical alignment of the cells in the left-hand column to the top of the cell using the command valign=”top” and the horizontal alignment to the left.

The cells of the right-hand column contain more text than those on the left and so the alignment of their contents will not be at the mercy of those to the left. In other words, they will fill all of the space they have been allocated and therefore it is safe for us to leave their alignment and vertical positioning at the default setting.

Creating a table by hand is often a time consuming and laborious task as at the end of the day the browser is pretty thick when it comes to handling data in this way. You cannot tell it to make all cells in the left-hand column blue, for instance, without telling it that on a cell-by-cell basis. Likewise, you cannot tell it to use a particular font or font size throughout. Again, this must be done on a cell-by-cell basis, hence the level of repetition in Fig 2. Let’s simplify matters. Although our table is only two cells wide by two cells deep, the above structure may seem a little confusing. So, ignore everything in the cells above and look at the bare bones structure of the table, which goes something like that shown in Fig 3.

And the moral of the above story? Define your rows, then your columns. Got that? Good. Now let’s take a look at forms.

What is a form?

Doubtless you’ve visited a web site and been asked to fill in a number of boxes and click ‘Submit’ to send the details to the owner of the site. What you filled in there was a form.

A form has two main functions: either, as in the above example, for collecting data to be sent back to the site administrator, or for interrogating a database even if that database contains only usernames and passwords so that you can gain access to a particular site.

[FIG3]

The bare bones

```
<table>
  <tr>
    <td>This is row 1,
column1</td>
    <td>This is row 1, column
2</td>
  </tr>
  <tr>
    <td>This is row 2, column
1</td>
    <td>This is
row 2, column
2</td>
  </tr>
</table>
```

It would be possible to write a whole chapter on forms but due to space constraints I'm going to have to condense it. As with almost everything to do with web design, the easiest way to learn is to look at what other people have done and see how the basic principles of their creations can be applied to your own site. To avoid infringing copyright, I'll direct you to my own site at www.nikzone.com (see Fig 4). Click on 'contact' for an example of a very simple form. Now view the source code using the option on the 'Edit' or 'View' menu of your browser. You'll be able to spot where the form begins and ends as it is defined by the <form> and </form> tags. The first of these tags also defines how the browser should handle the data it collects. In this instance, it is using a small Perl application — you'll be able to identify it by the .pl extension in the <form> definition — to 'POST' the data back to the site administrator's email address. Exactly how you do this will depend on your choice of ISP. Each differs in the way it likes its users to handle forms but most will have comprehensive instructions on the central web site, or be able to offer advice over the phone.

▲ Fig 4 You can use simple forms to gather data

A form is a collection of elements ranging from text boxes to drop-down lists and radio buttons. Every form, though, has at least one element in common; the 'Submit' button. Many also host a 'Reset' button that will clear the form should the user make a mistake. The two lines of text in Fig 5 define the submit and reset buttons. As you can see, they require no complex coding because when they are encountered by the browser it merely skips back to the opening <form> tag that defined how the form should be handled (in the case of the nikZone site, POSTed) to execute that command.

Adding elements

Let's add some more elements to our form. First, we want to know who is sending the data. As this is a general contact form we want to be able to reply to the emails it generates. To capture this data we'll insert a text box for the user to complete.

In the line of code in Fig 6 we define the form element as a text field (type="text") and name it "email". This name could be anything you like as it is merely a way of helping you identify what that line of data means when it arrives in your inbox. On screen, the box will be just 30 characters long (size="30") but it will allow the user to enter up to 75 characters by scrolling horizontally (maxlength="75"). All fine and well but what about the message itself? We don't want our visitors to have to enter this as one long line in a scrolling bar, so instead of using the 'text' input type we opt for the 'textarea' [Fig 7].

Following the logic of the 'text' input type, the 'textarea' line of code should be relatively easy to interpret. A number of subtle changes have occurred, however. 'Size' has become 'cols' and now we have a 'rows' parameter. Together they define this text box to be 45 characters wide and five rows deep. There will therefore be no horizontal scrolling, just vertical movement when the user reaches the bottom of the box. Wrapping has been set to 'VIRTUAL', so as the user reaches the right-hand margin the browser inserts a soft paragraph return which is stripped out before the data is returned to the site administrator. Setting this to 'PHYSICAL' would have put a hard return at the end of each line, and as the lines would never exceed 45 characters in length this would soon become tedious.

Now you have all the elements of your contact form it is a simple matter to arrange them in the order you feel most appropriate for your site. Normal convention is to have the email address first, the text area below it and the submit and reset buttons at the foot, but HTML allows you the freedom to work the way that suits you best.

So that concludes our quick look at web authoring by hand. In the three parts of this series we have only had a chance to look at the very basics of design and implementation. HTML is a simple, powerful and versatile language which because of its plain English format is easy to learn. If you have been following this series and it has inspired you to create a site of your own, do let me know and I'll have a peep. My contact details are below.

[FIG 6]

```
Email address <input type="text" name="email" size="30" maxlength="75">
```

[FIG 7]

```
<textarea name="textfield" cols="45" rows="5" wrap="VIRTUAL"></textarea>
```

[FIG5] Submit and reset buttons

```
<input type="submit" name="Submit" value="Submit">
<input type="reset" name="Reset" value="Reset">
```

PCW CONTACTS

Nik Rawlinson welcomes your comments. He can be contacted via the PCW editorial office (address, p14) or email nik_rawlinson@vnu.co.uk. His online home is www.nikzone.com.