Tables

A table is a grid of cells built of rows and columns. Originally, designers used tables to (predictably) display tables of information. But crafty Web developers quickly discovered that *invisible* tables offered a perfect way to lay out content in a variety of new ways (see Figure 9-2).

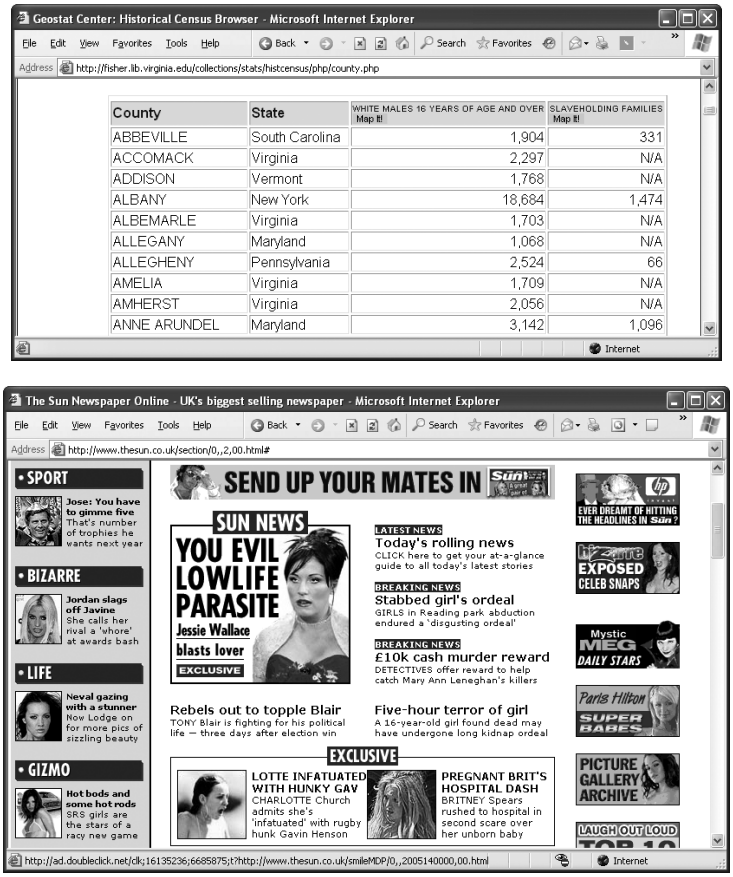


Figure 9-2:
Top: This detailed census information from 1790 makes perfect sense in an ordinary table.

Bottom: A combination of invisible tables (technically, tables with no borders) gives you all the underpinning you need for this headache-inspiring, multi-columned newspaper view.

In the following sections, you'll explore how to create tables using XHTML.

The Anatomy of a Table

You can whip up a table with just a few new XHTML elements:

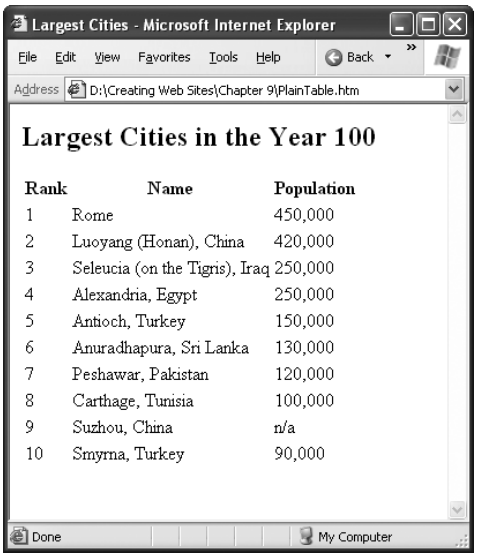
- **<table>** wraps the whole shebang. It's the starting point for every table.
- **<tr>** represents a single table row. Every table element (<table>) contains a series of one or more <tr> elements.

- **<td>** represents a table cell (it stands for table data). For each cell you want in a row, you add one <td> element. You put the text (or numbers, or <img> elements, or pretty much any XHTML you like) that you want to appear in that cell inside the <td> element. If you put text in the cell, it gets displayed in the same font as ordinary body text.
- **<th>** is an optional table element you use when you want to define a column heading. You can use a <th> element instead of a <td> element any time, although it usually makes the most sense in the first row of a table. Browsers format the text inside the <th> element in almost the same way as text in a <td> element, except that they automatically boldface and center the text (unless you apply different formatting rules with a style sheet).

Note: There are a few other table-specific elements, but they've fallen by the wayside. These elements, which designers no longer need or that browsers don't universally support, include <thead>, <tbody>, <tfoot>, and <caption> elements.

Figure 9-3 shows a table at its simplest. Here's a portion of the XHTML used to create that table:

```
<table>
  <tr>
    <th>Rank</th>
    <th>Name</th>
    <th>Population</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Rome</td>
    <td>450,000</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Luoyang (Honan), China</td>
    <td>420,000</td>
  </tr>
  <tr>
    <td>3</td>
    <td>Seleucia (on the Tigris), Iraq</td>
    <td>250,000</td>
  </tr>
  ...
</table>
```



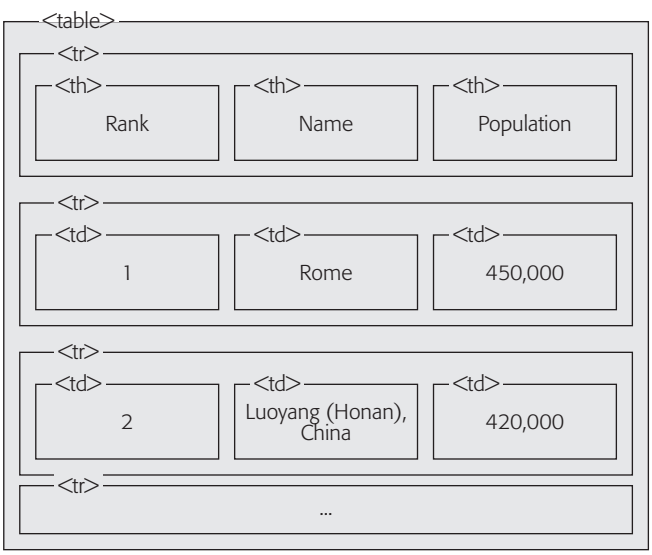


Figure 9-3:
Top: This basic table doesn't have any borders thanks to a style sheet rule, but you'll still spot the signature sign that you're looking at a table: text lined up neatly in rows and columns.

Bottom: This behind-the-scenes look at the XHTML powering the table above shows the <table>, <tr>, <th>, and <td> elements for the first three rows.

The markup for this table uses indented table elements to help you see the structure of the table. Indenting table elements like this is always a good idea, as it helps you spot mismatched tags. In this example, the only content in the <td> elements is ordinary text. But you can put other XHTML elements in cells, too, including hyperlinks (the <a> element) and images (the <img> element).

Tip: You might be able to avoid writing tables by hand, as most Web design tools include their own table editors that let you point and click your way to success. These table-creation features are similar to those you find in a word processor.

Formatting Table Borders

Traditional tables have borders around each cell. You can adjust those table borders using the *border* attribute. It specifies the width (in pixels) of the line that browsers add around each cell and around the table as a whole. Here's an example:

```
<table border="1">
  ...
</table>
```

Although you can choose the border's line thickness, you can't control its style. Most browsers outline a table using a solid black line with a raised edge (see Figure 9-4).

Don't feel limited by these automatic borders. If you don't like them, there's a work-around: style sheets. The basic trick is to create a borderless table, and then apply a border to some combination of the <tr>, <td>, <th>, and <table> elements. You can find style sheet border properties described on page 167.

The following style sheet rules set a thin blue border around every cell, and a thick blue border around the table itself:

```
table {
  border-width: 3px;
  border-style: solid;
  border-color: blue;
}
td, th {
  border-width: 1px;
  border-style: solid;
  border-color: blue;
}
```

Figure 9-4 shows the result.

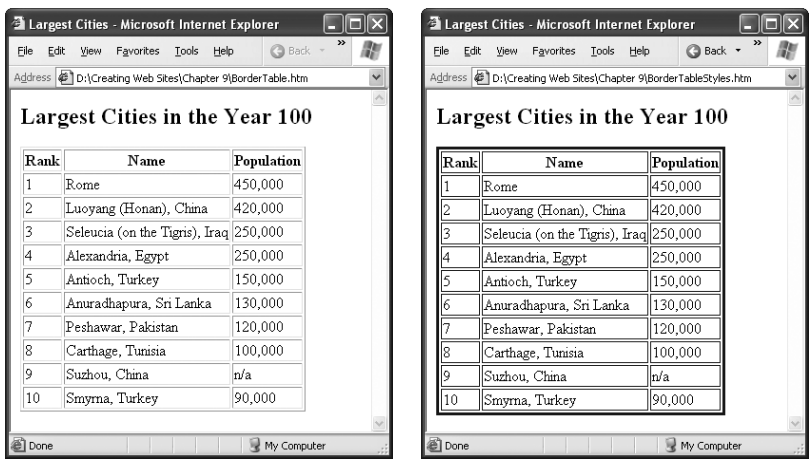


Figure 9-4:
Left: A standard XHTML table border with a thickness of 1 pixel.

Right: A custom border using style rules.

Tip: Borders aren't the only style sheet feature you can apply to table cells. You can also change a cell's text font and text alignment (page 153), its padding and margins (page 163), and the colors it uses (page 151). You can even set a background image for an individual cell or the whole table using the background-image property (page 198). And if you want to apply style rules to individual cells (rather than to the table as a whole), you just need to use class names (page 171).

There's one hiccup to watch for when you create tables with borders. If the table includes empty cells, they'll appear "collapsed," which means they won't get any borders at all (see Figure 9-5).

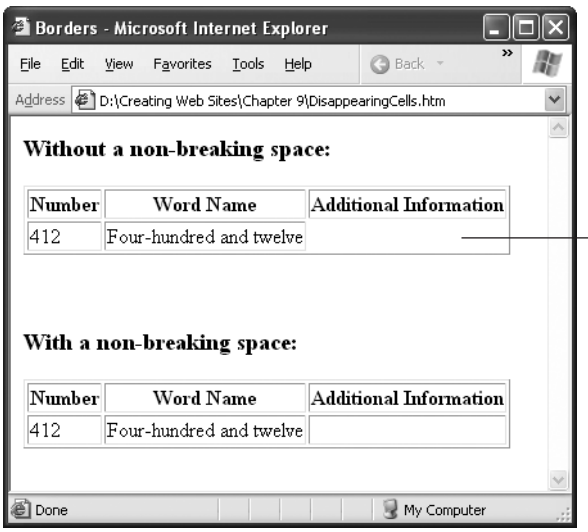


Figure 9-5:
If you don't include a non-breaking space, you'll lose the borders around empty cells.

To prevent cells from collapsing, add a single non-breaking space to them:

```
<td> </td>
```

The browser won't display this space, but it will ensure that your borders stay put.

Cell Spans

Tables support *spanning*, a feature that lets a single cell stretch out over several columns or rows. Think of spanning as XHTML's version of the Merge Cells feature in Microsoft Word and Excel.

Spanned cells let you tweak your tables in all kinds of funky ways. You can, for example, specify a *column span* to stretch a cell over two or more columns. Just add a *colspan* attribute to the <td> element you want to extend, and specify the total number of columns you want to skip.

Here's an example that stretches a cell over two columns, so that the cell ends up occupying the space of two full cells:

```
<table>
  <tr>
    <td>Column 1</td>
    <td>Column 2</td>
    <td>Column 3</td>
    <td>Column 4</td>
  </tr>
  <tr>
    <td> </td>
    <td colspan="2">Look out, this cell spans two columns!</td>
    <td> </td>
  </tr>
  ...
</table>
```

Figure 9-6 shows this trick in action.

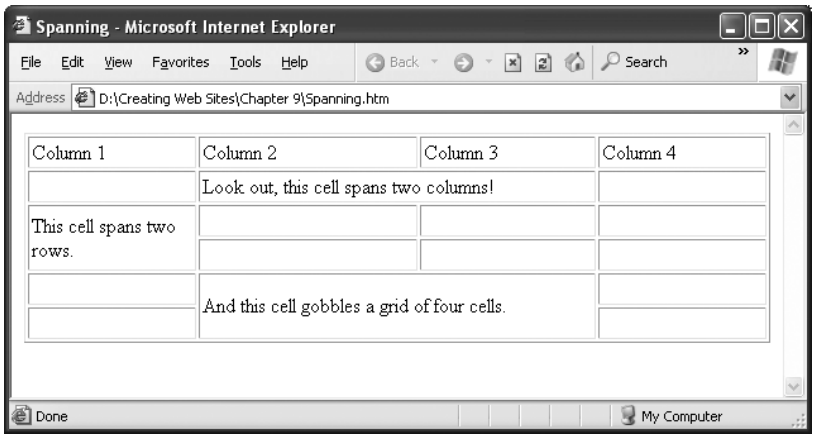


Figure 9-6: *A table with row spanning and column spanning run amok.*

To make sure your table doesn't get mangled, you need to keep track of the total number of columns you have to work with in each row. In the previous example, the first row starts off by defining four basic columns:

```
<tr>
  <td>Column 1</td>
  <td>Column 2</td>
  <td>Column 3</td>
  <td>Column 4</td>
</tr>
```

In the next row, the second column extends over the third column, thanks to column spanning (see the markup below). As a result, the third <td> element actually

represents the *fourth* column. That means you need only three <td> elements to fill up the full width of the table:

```
<tr>
  <!-- This fills column 1 -->
  <td> </td>

  <!-- This fills columns 2 and 3 -->
  <td colspan="2">Look out, this cell spans two columns!</td>

  <!-- This fills column 4 -->
  <td> </td>
</tr>
```

This same principle applies to *row spanning* and the *rowspan* attribute. In the following example, the first cell in the row leaks through to the second row:

```
<tr>
  <td rowspan="2">This cell spans two rows.</td>
  <td> </td>
  <td> </td>
  <td> </td>
</tr>
```

In the next row, the cell from above already occupies the first cell, so the first <td> element you declare actually applies to the second column. All in all, therefore, this row needs only three <td> elements:

```
<tr>
  <td> </td>
  <td> </td>
  <td> </td>
</tr>
```

If you miscount and add too many cells to a row, you end up with an extra column at the end of your table.

Tip: Many Web page editors let you create spans by joining cells. In Dreamweaver and Expression Web, select a group of cells, right-click them, and then select Merge Cells.

Sizing and Aligning Tables

If you don't explicitly set the size of a table, it gets as wide as necessary to display all its columns, and each column grows just wide enough to fit the longest line of text (or to accommodate any other content you've added, like a picture by adding an <img> element). However, there's one additional rule—the table can't grow wider than the browser window. Once a table reaches the full width of the current window, the browser starts wrapping the text inside each column, so that the table grows taller as you pile in more content.

Tip: Need more space inside your table? Style rules can make it easy. To add more space between the cell content and its borders, increase the padding property for the <td> and <tr> elements. To add more space between the cell borders and any adjacent cells, up the margin width for the <td> and <tr> elements. Page 163 has more on adjusting these dimensions.

In most cases, you want to explicitly set the width of your table and its individual columns. When you do so, the table respects these dimensions and wraps text to accommodate those widths. Once again, style sheets provide the best approach. All you do is set height and width properties, as explained in the next section.

Sizing a table

You have two choices when you specify table dimensions:

- **Relative sizing** sizes a table in sync with the dimensions of a browser window. You supply the percentage of the window you want the table to fill.
- **Absolute sizing** uses pixel measurements to set the exact size of a table.

The following style sheet rule ensures that this table always occupies the full width of a browser window:

```
table {
  width: 100%;
}
```

A browser sizes this table *in relation to* the size of the browser window.

This rule limits the table to half the width of the current window:

```
table {
  width: 50%;
}
```

Either way, the table dynamically resizes as you resize the browser window.

Note: In the examples you've seen so far, all the tables have been plopped directly into the <body> section of a page. This is a common design, but it's not the only possibility–alternatively, you can put a table inside some other container element. If you use a relative width for this table, its width is proportional to the size of its container. So if you set the table width to 50%, it takes half the width of its container, no matter how big or how small it is. To try this out, put your table in a cell inside another table (a trick described on page 247) or in a element that uses a fixed width (page 253).

If you use exact pixel widths, the table dimensions never change. For example, the following rule creates a table that's a generous 500 pixels wide:

```
table.Cities {
  width: 500px;
}
```

Because you create this table with a specific width, you should only use it for content that justifies that width. To prevent your browser from applying this style to every table it encounters (because the table's size may not suit every type of content), you give it a class name, like Cities in the example above. To display this specific table, you cite the table class in your XHTML document:

```
<table class="Cities">
    ...
</table>
```

In addition to specifying the width of a table, you can also set its height. Usually, you set the height using an absolute size, as shown here:

```
table.Cities {
  height: 500px;
}
```

That's because, if you set a table height as a percentage of a browser window, it creates a strange (and rarely seen) effect—the table grows taller and shorter as you lengthen or shorten your browser window.

There's one important caveat to table sizing. Although you can make a table as large as you want (even if it stretches beyond the borders of a browser window), you don't have the same ability to shrink a table. If you specify a table size that's smaller than the minimum size the table needs to display your content, the table appears at this minimum size (see Figure 9-7).

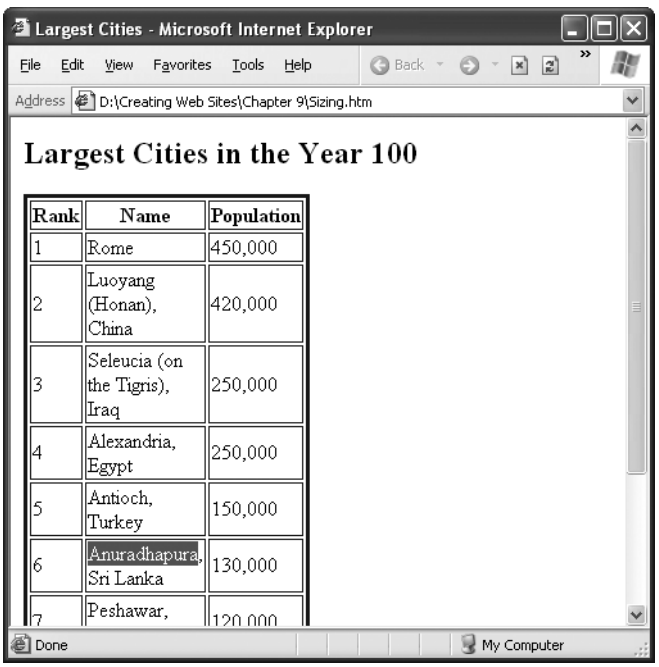


Figure 9-7:
In this example, the XHTML called for a table width of 1 pixel. But the browser doesn't shrink the table down that far because the content influences the table's minimum size. In this table, the city name Anuradhapura is the longest un-splittable value, so the browser uses that name to determine the width of the column. If you really want to ratchet the size down another notch, try shrinking the text by applying a smaller font size.

Sizing a column

Now that you know how to size a table, you probably want to know what your browser does if a table has more than enough space for its content. Once a table reaches its minimum size (just large enough to fit all its data), your browser distributes any extra space proportionately, so that every column width increases by the same amount.

Of course, this isn't necessarily what you want. You might want to create a wide descriptive column paired with a narrow column of densely packed text. Or you might want to set columns to a specific size so that all your pages look uniform, even if the content differs.

To set a column's size, you use the *width* property in conjunction with the <td> and <th> elements. Once again, you can do this proportionately, using a percentage, or exactly, using a pixel width. However, proportional sizing has a slightly different effect when you use it with columns. Earlier, when you used a percentage value for table width, you sized the entire table relative to the width of the page. In that example, you had a table width of 50%, which means the table occupied 50 percent of the full width of the page. But when you use a percentage value to set a *column* width, you're defining the percentage of the *table* width that the column should occupy. So when you set a column width of 50%, the column takes up 50 percent of the *table*.

When you size columns, you need to create a style rule for each one, giving it a unique class name (page 171). That's because each column is potentially a different width—you can't just write a single style rule that applies to every column, unless you want them all to have exactly the same width.

The following style rules set different widths for each column of the table you saw in Figure 9-7.

```
th.Rank {
  width: 10%;
}
th.Name {
  width: 80%;
}
th.Population {
  width: 10%;
}
```

In this example, the class names match the column titles, which makes it easy to keep track of which rule applies to which column.

Tip: When you use percentage widths for columns, you don't need to specify values for all three columns. If you leave one out, the browser sizes that column to fill the rest of the space in the table. If you do decide to include widths for each column (as in the previous example), make sure they add up to 100 percent to avoid confusion. Otherwise, the browser will override one of your settings, and you won't know how your table will actually appear.

For these rules to take effect, you need to apply them to the corresponding cells:

```
<table class="Cities">
  <tr>
    <th class="Rank">Rank</th>
    <th class="Name">Name</th>
    <th class="Population">Population</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Rome</td>
    <td>450,000</td>
  </tr>
  ...
</table>
```

Notice that you specify widths only for the column elements in the first row (the ones that contain the cell headers in this example). You could apply the rule to every row, but there's really no point. When the browser builds a table, it scans the whole table structure to determine the required size, based on the cell content and any explicit width settings. If you apply a different width to more than one cell in the same column, the browser simply uses the largest value.

Tip: It's a good idea to size your table by applying style rules to the first row. This makes your XHTML more readable, because it's immediately obvious what the dimensions of your table are.

Sizing a row

You can size a row just as easily as you size a column. The best approach is to use the *height* property in the <tr> attribute, as shown here:

```
tr.TallRow {
  height: 100px;
}
```

Once again, XHTML lets you use either percentages or pixel values. When you resize a row, you affect every cell in every column of that row. However, you're free to make each row in the table a different height, using the techniques just described.

Organizing a Page with Tables

So far, the tables you've seen have been fairly typical grids of information. But on many Web sites, tables play another role—they organize pages into separate regions of content.

One of the most common Web site designs is to divide a page into two or three columns. The column on the left typically holds navigation buttons or other links.

The column in the middle takes up the most space and includes the main content for the page. The column on the right, if present, displays additional information, like an advertisement or another set of links. Figure 9-8 shows how it all breaks down.

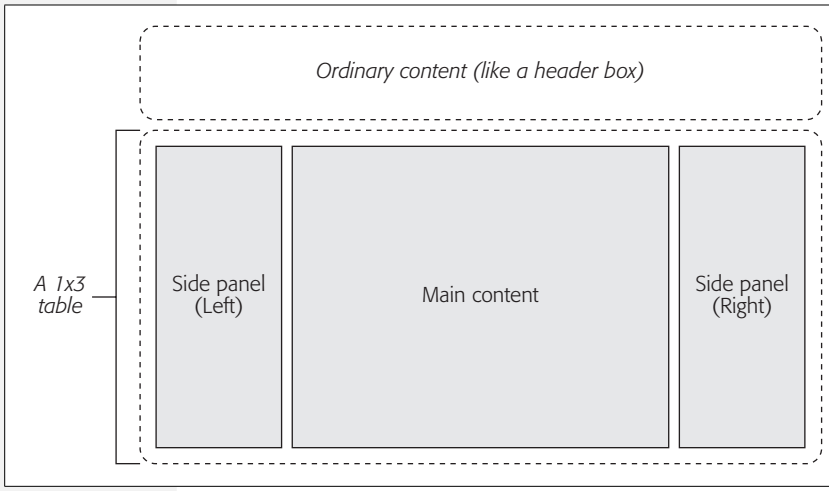


Figure 9-8: *A simple table with one row and three columns is all you need to define the overall structure of this Web page. All your content can fit into a single large row. Most visitors won't realize there's a table here at all.*

To make this design work, you need to consider several details:

- **Vertical text alignment**. Ordinarily, browsers position row content in the center of the row, between the top and bottom edge. This isn't what you want in tall rows, and it's a particularly bad choice for any row that's more than a screenful high. For example, if you have a set of navigation links in the left-most column and that column extends below the bottom of the screen, your vertically centered links could fall below the bottom of the screen, too. To ensure this doesn't happen, you want to have your browser put the content in each cell at the top of the table, so it sits at the top of your page, immediately visible.
- **Borders**. If you decide to use borders, you'll want them only on *some* edges to emphasize the separation of content. You won't want them around every cell or around the entire table. In many cases, you'll do away with borders altogether and use different background colors or images to separate the sections of a page.
- **Sizing**. Typically, you set a specific width for the left and right side panels. The middle panel needs to command the most space. If you leave its width unspecified, it can grow as visitors enlarge their browser windows or as they switch to widescreen monitors.

Figure 9-9 gives you a taste of what a finished page that uses a table to lay out the page might look like. You can see the entire page with the downloadable content for this chapter (go to *www.missingmanuals.com*, and then click the Missing CD link).

The table in this example is relatively simple because you keep and maintain all its formatting and sizing details separately, in a style sheet. Therefore, all you need to

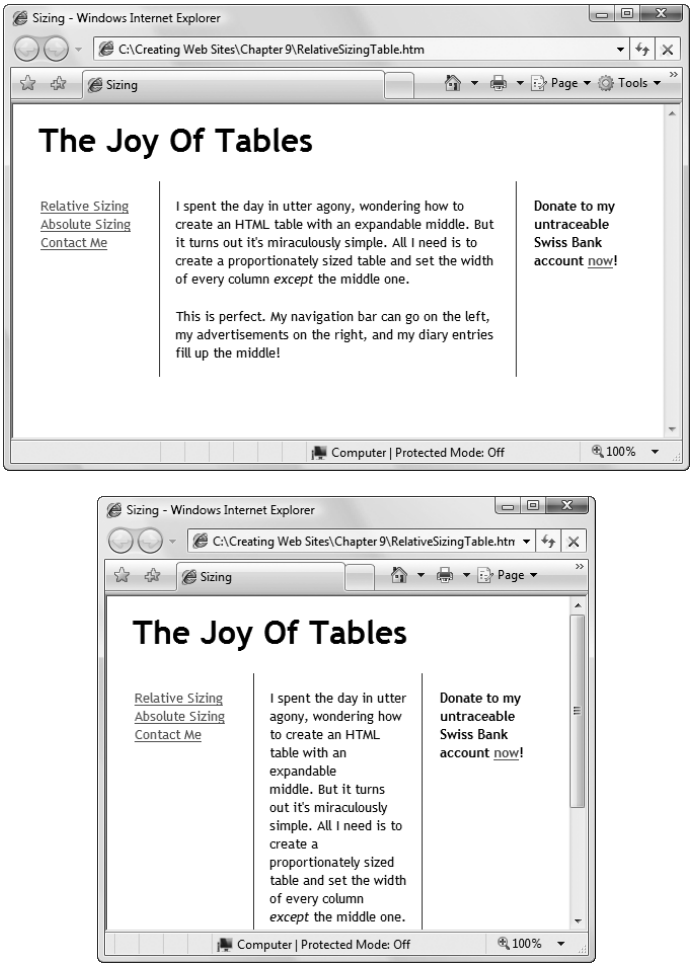


Figure 9-9:
Top: A relatively easy-to-make table creates this attractive page layout.

Bottom: When you shrink the size of the browser window, the side panels remain the same size; only the middle panel changes.

do is create an ordinary table with one row and three columns. You then map each column to a different style using a class name:

```
<table>
  <tr>
    <td class="Left">
      <a href="RelativeSizing.htm">Relative Sizing</a><br />
      <a href="AbsoluteSizing.htm">Absolute Sizing</a><br />
      <a href="mailto:no-one">Contact Me</a><br />
    </td>
    <td class="Middle">
      I spent the day in utter agony, wondering how
      to create a table with an expandable middle...
    </td>
```

```
    <td class="Right">
      Donate to my untraceable Swiss Bank account
      <a href="http://www.paypal.org">now</a>!
    </td>
  </tr>
</table>
```

The style sheet rules start by specifying a font for the whole document:

```
body {
  font-family: Trebuchet MS, serif;
}
```

Next, you size the table to fill the browser window:

```
table {
  width: 100%;
}
```

Then you give every cell some standard settings for text alignment, font size, and padding (to provide a little extra space between the column border and the text):

```
td {
  font-size: x-small;
  padding: 15px;
  vertical-align: top;
}
```

Finally, you set the widths and borders with column-specific rules. Here are the two rules that give the side panels fixed, 100-pixel widths:

```
td.Left {
  width: 100px;
}
td.Right {
 width: 100px;
 font-weight: bold;
}
```

Next up is the style sheet rule for the middle column. Unlike the side columns, the middle column doesn't have an explicit width. Instead, the browser sizes it to fit whatever space remains. This style sheet rule also gives the middle column left and right borders to separate it from the side panels:

```
td.Middle {
  border-left-width: 1px;
  border-right-width: 1px;
  border-top-width: 0px;
  border-bottom-width: 0px;
  border-style: solid;
  border-color: blue;
}
```

There's one last detail you might want to change in the above style sheet. This example uses proportional sizing for the table, which lets the middle panel grow and shrink as visitors resize their browser window. Although this is the most flexible option, in dense, graphics-rich Web sites where you've precisely positioned text, images, sidebars, and other content, you may need absolute sizing to preserve your carefully crafted layout. (For more about this issue, see page 230.)

You can convert the above example to use absolute sizing by changing the style rule that applies to the <table> element, as shown here:

```
table {
  width: 600px;
}
```

This sets the table width at 600 pixels. The left and right panels are still 100 pixels wide, so the middle column gets whatever's left—in this case, 400 pixels (based on a total width of 600 pixels, minus 100 pixels for each panel). Figure 9-10 shows the difference.

POWER USERS' CLINIC

Nested Tables

In sophisticated Web sites, the show doesn't end with a single table. Instead, site creators put tables *inside* other tables, which they then put inside yet more tables. For example, you might create a basic three-column setup using one table, and then divide the right column into a series of distinct ads using a second table. When you put one table inside another like this, you create a *nested table*.

Building nested tables is easy, although it can be a little difficult to keep track of everything. The trick is to define a table inside one of the cells in an existing table. For example, if you have this table with three columns:

```
<table>
  <tr>
    <td class="Left">...</td>
    <td class="Middle">...</td>
    <td class="Right">...</td>
  </tr>
</table>
```

You can slide a table right into the <td> tags for the third column:

```
<table>
  <tr>
    <td class="Left">...</td>
    <td class="Middle">...</td>
    <td class="Right">
      <table>
        <tr>...</tr>
        <tr>...</tr>
      </table>
    </td>
  </tr>
</table>
```

Resizing all the parts of these two tables can get confusing. It's easiest to size the nested table using a relative width of 100 percent. That way, the nested table expands or shrinks based on the width of the column it's in.

Style-Based Layout

Although the table-based approach to page layout seems perfect at first, it has a few frustrating quirks. One of the most daunting is that once you perfect your table-based layout, you need to painstakingly copy the exact table structure to every