

# 4

## Tables

Tables display information in rows and columns; they are commonly used to display all manner of data that fits in a grid such as train schedules, television listings, financial reports, and sports results. In this chapter, you learn when to use tables, and the markup that you need to create them.

To begin this chapter, we'll look at some examples of tables, then quickly move onto the basic elements that are used to create them. Having learned the basics, you can then go on to learn some of the more advanced features of tables such as adding captions and headings, and how to achieve more complicated table layouts. Along the way, you will also learn some deprecated markup that was designed to control the appearance of tables because, even though it is preferable to use CSS to control the way a page looks, you may sometimes come across pages that use the older markup.

The chapter ends with a discussion of accessibility issues that relate to tables, because it is important to understand how a screen reader would read the contents of a table to users with visual impairments.

### Introducing Tables

In order to work with tables, you need to start thinking in *grids*, so let's start off by looking at some examples of how popular web sites use tables.

# Chapter 4: Tables

In Figure 4-1 you can see the NFL web site. This page shows the standings for each team in a table. You can see a list of teams down the left, and for each team there are columns providing different stats, including number of games won, lost, or tied.

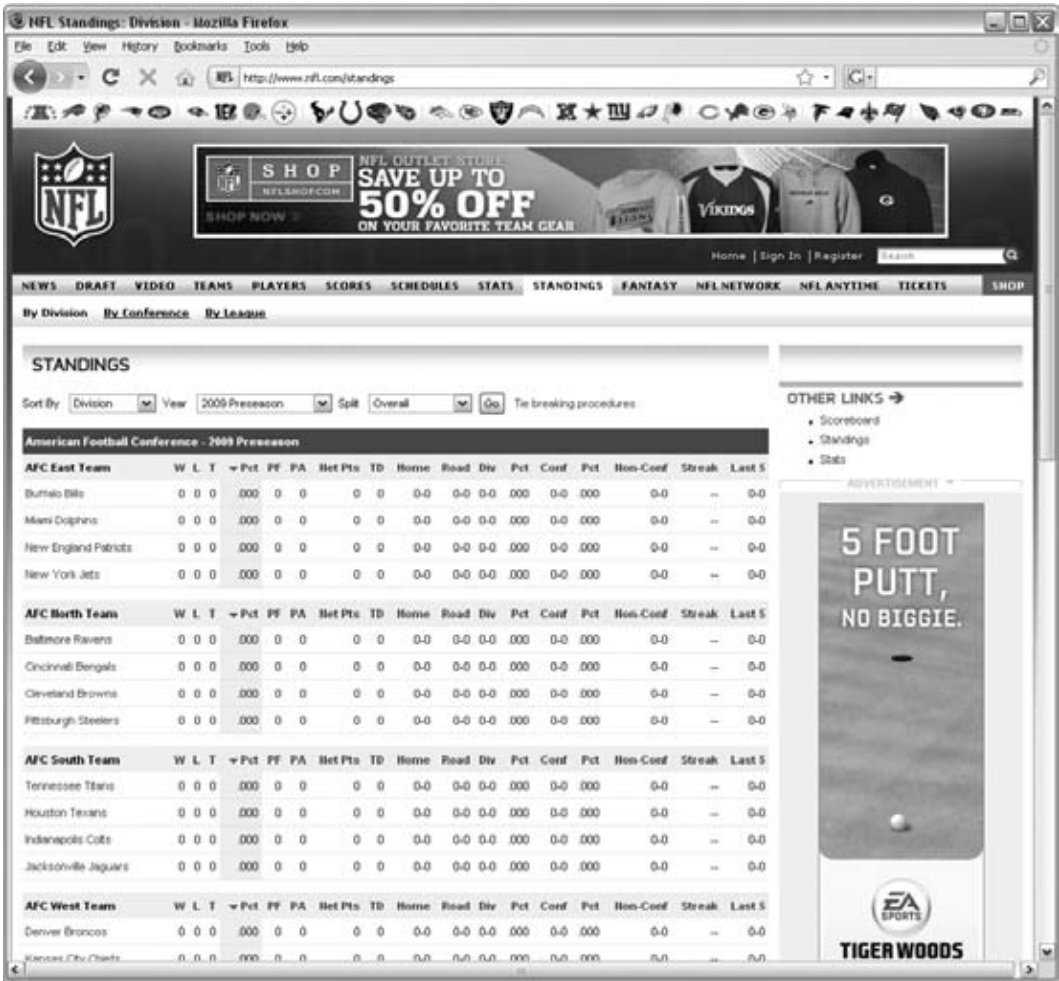


Figure 4-1

Figure 4-2 shows the Bloomberg web site. This page shows major stock markets. In this case, there are tables for different regions around the world (in this screenshot, you can see North and Latin America as

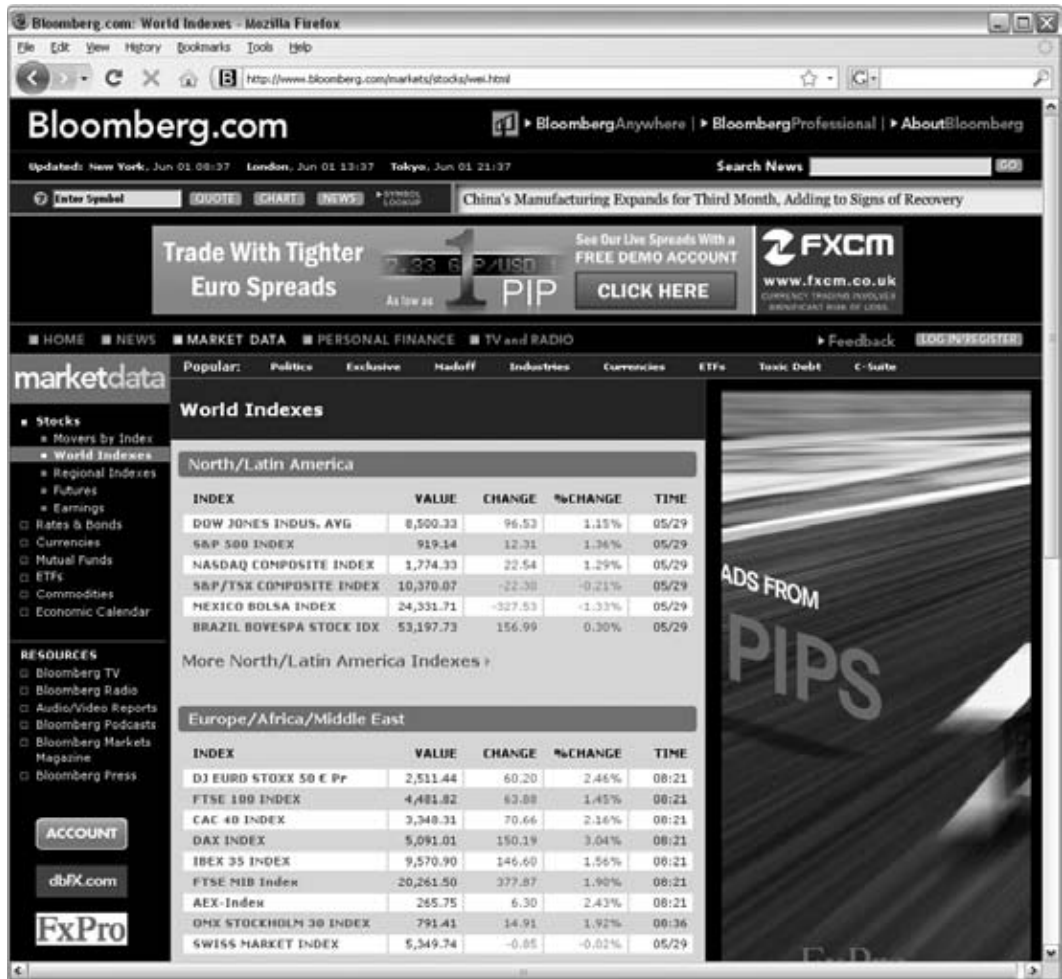


Figure 4-2

the first region, followed by Europe, Africa, and the Middle East). Each table contains the major indexes trading in that region down the left-hand side, followed by columns that show the current value, fluctuations in values, and date/time of the stats.

# Chapter 4: Tables

In Figure 4-3 you can see the web site for the Heathrow Express, a train between Central London and London’s Heathrow Airport. The tables on this page show the times that the trains run.

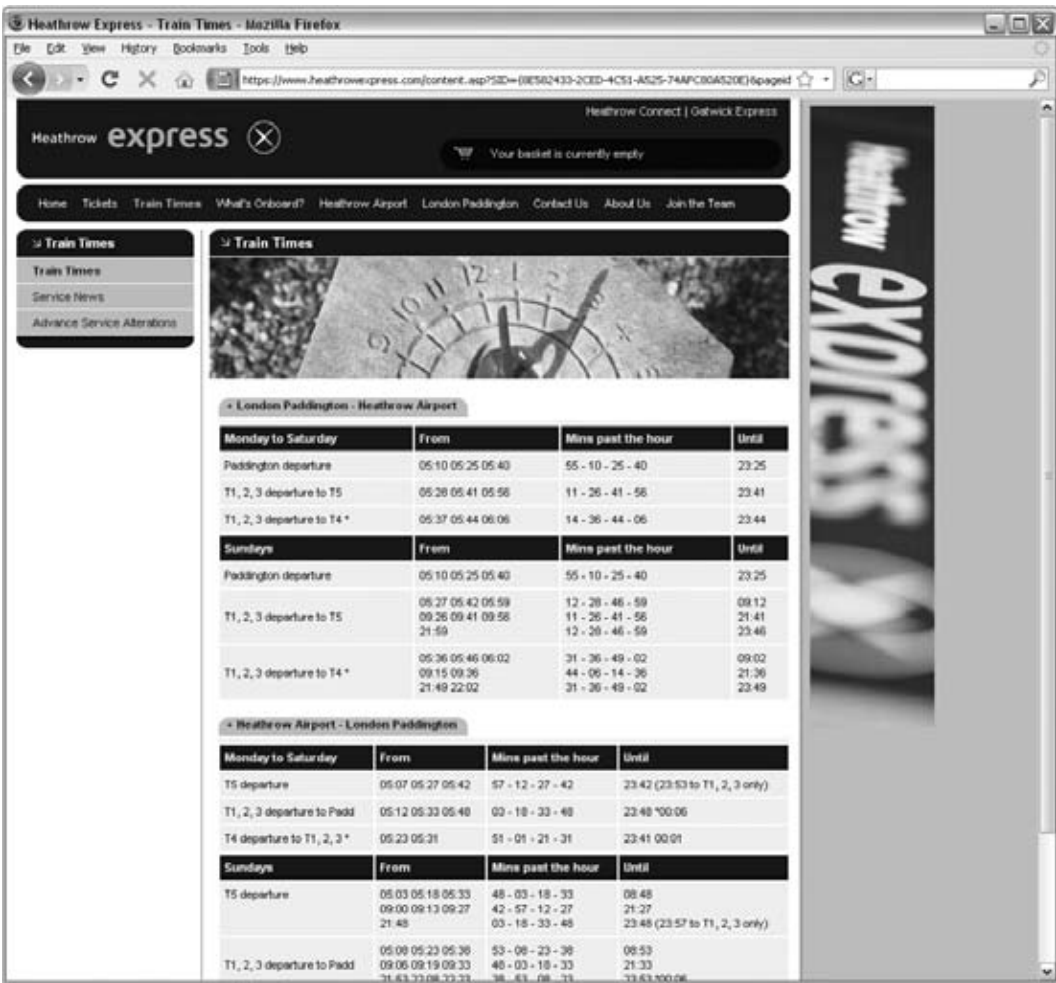


Figure 4-3

In Figure 4-4, you can see the *New York Times* web site. On this page, the tables provide television listings.



Figure 4-4

I hope these examples give you a better idea of what a table is and when you might want to use one. I am not suggesting that every web site that displays stock market data or television listings should use a table — rather that you should consider using a table when you need to display information that sits well in a grid of rows and columns. If you are looking at a web page and want to know whether that page is using a table to control how the data is laid out, you can always look at the source for that page and look for the elements you are about to read about in this chapter.

# Chapter 4: Tables

You can think of a table as being very similar to a spreadsheet because it is made up of rows and columns, as shown in Figure 4-5.

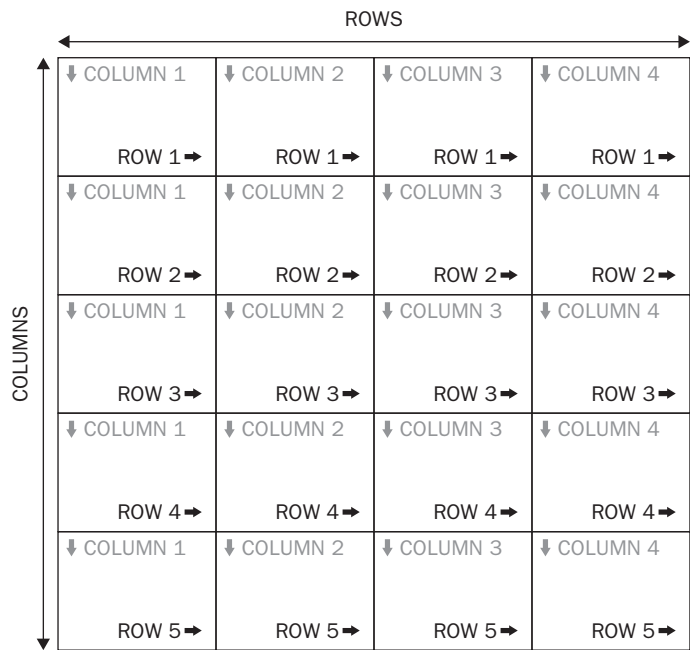


Figure 4-5

Here you can see a grid of rectangles. Each rectangle is known as a *cell*. A *row* is made up of a set of cells on the same line from left to right, and a *column* is made up of a line of cells going from top to bottom.

Let’s look at an example of a very basic XHTML table so that you can see how they are created (see Figure 4-6).

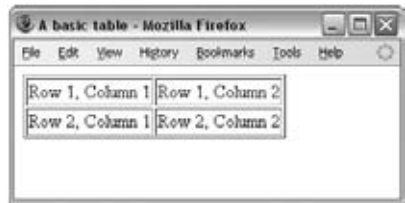


Figure 4-6

You create a table in XHTML using the `<table>` element. Inside the `<table>` element, the table is written out row by row. A row is contained inside a `<tr>` element — which stands for *table row*. Each cell is then written inside the row element using a `<td>` element — which stands for *table data*.

Here is the code that was used to create this basic table (ch04\_eg01.html):

```
<table border="1">
  <tr>
    <td>Row 1, Column 1</td>
    <td>Row 1, Column 2</td>
  </tr>
  <tr>
    <td>Row 2, Column 1</td>
    <td>Row 2, Column 2</td>
  </tr>
</table>
```

When writing code for a table in a text editor, I'm always extra careful to start each row and cell on a new line and to indent table cells inside table rows as shown. If you are using a web page authoring tool such as Dreamweaver, it will probably automatically indent the code for you.

Many web page authors find it particularly helpful to indent the code for a table because leaving off just one tag in a table can prevent the entire table from being displayed properly. Indenting the code makes it easier to keep track of the opening and closing of each element.

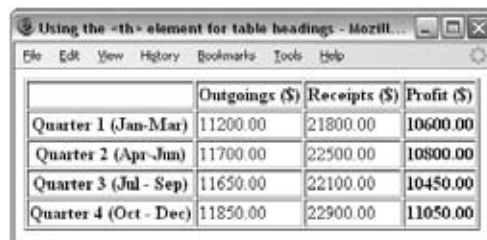
Take a look at the same code again. This time, it has not been split onto separate lines or indented, and I'm sure you will agree it's much harder to read.

```
<table border="1"><tr><td>Row 1, Column 1</td><td>Row 1, Column 2</td></tr><tr>
<td>Row 2, Column 1</td><td>Row 2, Column 2</td></tr></table>
```

All tables will follow this basic structure, although there are additional elements and attributes that allow you to control the presentation of tables. If a row or column should contain a heading, a `<th>` element should be used in place of the `<td>` element for the cells that contain a heading. By default, most browsers render the content of a `<th>` element in bold text.

**Each cell must be represented by either a `<td>` or a `<th>` element in order for the table to display correctly, even if there is no data in that cell.**

Let's take a look at a slightly more complicated table (see Figure 4-7). This time the table includes headings. In this example, the table shows a financial summary for a small company.



|                     | Outgoings (\$) | Receipts (\$) | Profit (\$) |
|---------------------|----------------|---------------|-------------|
| Quarter 1 (Jan-Mar) | 11200.00       | 21800.00      | 10600.00    |
| Quarter 2 (Apr-Jun) | 11700.00       | 22500.00      | 10800.00    |
| Quarter 3 (Jul-Sep) | 11650.00       | 22100.00      | 10450.00    |
| Quarter 4 (Oct-Dec) | 11850.00       | 22900.00      | 11050.00    |

Figure 4-7

## Chapter 4: Tables

---

Here is the code that was used to create this table (ch04\_eg02.html):

```
<table border="1">
  <tr>
    <td></td>
    <th>Outgoings ($)</th>
    <th>Receipts ($)</th>
    <th>Profit ($)</th>
  </tr>
  <tr>
    <th>Quarter 1 (Jan-Mar)</th>
    <td>11200.00</td>
    <td>21800.00</td>
    <td><b>10600.00</b></td>
  </tr>
  <tr>
    <th>Quarter 2 (Apr-Jun)</th>
    <td>11700.00</td>
    <td>22500.00</td>
    <td><b>10800.00</b></td>
  </tr>
  <tr>
    <th>Quarter 3 (Jul - Sep)</th>
    <td>11650.00</td>
    <td>22100.00</td>
    <td><b>10450.00</b></td>
  </tr>
  <tr>
    <th>Quarter 4 (Oct - Dec)</th>
    <td>11850.00</td>
    <td>22900.00</td>
    <td><b>11050.00</b></td>
  </tr>
</table>
```

The first row is made entirely of headings for outgoings, receipts, and profit. Note how the top-left cell in Figure 4-7 is empty; in the code for the table, we still need an empty `<td>` element to tell the browser that this cell is empty (otherwise it has no way of knowing that there is an empty cell here).

In each row, the first cell is also a table heading cell (indicated using a `<th>`), which states which quarter the results are for. Then the remaining three cells of each row contain table data and are contained inside the `<td>` elements.

The figures showing the profit (in the right-hand column) are contained within a `<b>` element, which shows the profit figures in a bold typeface. This demonstrates how any cell can, in fact, contain all manner of markup. The only constraint on placing markup inside a table is that it must nest within the table cell element (be that a `<td>` or a `<th>` element). You cannot have an opening tag for an element inside a table cell and a closing tag outside that cell — or vice versa.



When creating tables, many people do not actually bother with the `<th>` element and instead use the `<td>` element for every cell — including headers. You should, however, aim to use the `<th>` element whenever you have a table heading.

*As you can see from the examples so far in this chapter, tables can take up a lot of space and make a document longer, but clear formatting of tables makes it much easier to see what is going on in your code. No matter how familiar the code looks when you write it, you will be glad that you made good use of structure if you have to come back to it a year later.*

## Basic Table Elements and Attributes

Now that you’ve seen how basic tables work, this section describes the elements in a little more detail, introducing the attributes they can carry. Some of the attributes allow you to create more sophisticated table layouts. Feel free to skim through this section fairly quickly; once you know what is possible to do with the markup, you can always come back again and study the markup more closely in order to see how to achieve what you want.

### The `<table>` Element Creates a Table

The `<table>` element is the containing element for all tables. It can carry the following attributes:

- ❑ All the universal attributes
- ❑ Basic event attributes for scripting

The `<table>` element can carry the following deprecated attributes. Even though they are deprecated, you will still see many of them in use today:

`align bgcolor border cellpadding cellspacing dir frame rules summary width`

### The `align` Attribute (Deprecated)

Although it is deprecated, the `align` attribute is still frequently used with tables. When used with the `<table>` element, it indicates whether the table should be aligned to the `left` (the default), `right`, or `center` of the page. (When used with cells, as you will see shortly, it aligns the content of that cell.) You use the attribute like so:

```
<table align="center">
```

You can put a whole table inside a single cell of another table (or inside other block-level elements), and if a table is contained within another element, the `align` attribute will indicate whether the table should be aligned to the left, right, or center of that element.

# Chapter 4: Tables

When the align attribute is used on a table, text should flow around it (rather like it can flow around an image). For example, here is a left-aligned table that is followed by some text (ch04\_eg03 .html):

```
<table border="1" align="left">
  <tr>
    <td>Row 1, Column 1</td>
    <td>Row 1, Column 2</td>
  </tr>
  <tr>
    <td>Row 2, Column 1</td>
    <td>Row 2, Column 2</td>
  </tr>
</table>
Lorem ipsum dolor sit amet, consectetur adipiscing elit...
```

The text should flow around the table, as shown in the first table in Figure 4-8.

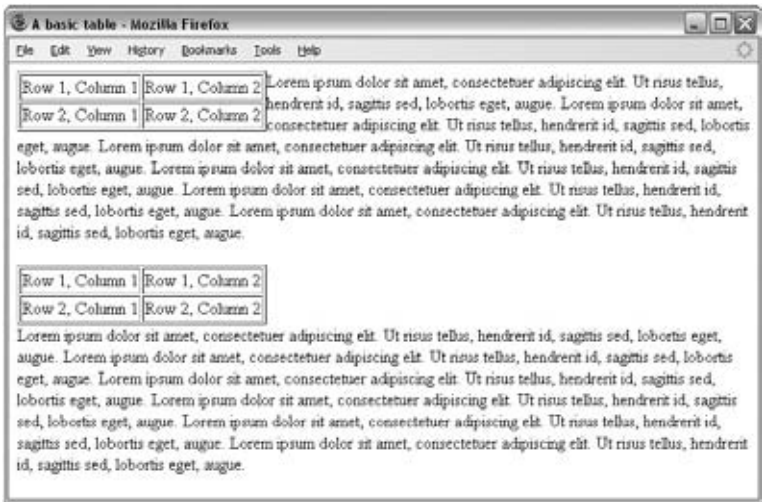


Figure 4-8

If you do not want the text to flow around the table, you can use the CSS clear property on the element after it (you will learn about this in Chapter 8). Alternatively, you *could* place a line break after the table and add the clear attribute, which is also deprecated `<br clear="left" />` (although the CSS clear property would be the preferred method). This is demonstrated in the second table of Figure 4-8.

```
</table>
<br clear="left" />
Lorem ipsum dolor sit amet, consectetur adipiscing elit...
```

The `clear` attribute indicates how the browser should display the next line after the line break. With the value of `left`, the text can begin only when there is nothing positioned on the left margin of the browser window (or containing element). The `clear` attribute is covered in more detail in Appendix I.

### ***The bgcolor Attribute (Deprecated)***

The `bgcolor` attribute sets the background color for the table. The value of this attribute should either be a color name or a six-digit code known as a *hex code*. The way in which colors are specified in XHTML and CSS is covered in Appendix D. (The `bgcolor` attribute has been replaced by the `background-color` property in CSS.)

The syntax is as follows:

```
bgcolor="red"
```

### ***The border Attribute (Deprecated)***

If you use the `border` attribute, a border will be created around both the table and each individual cell. The value for this attribute is the width you want the outside border of the table to be, in pixels. If you give this attribute a value of 0, or if you do not use this attribute, then you should not get any borders on either the table or any cells. (This has been replaced by the `border` property in CSS.)

```
border="0"
```

While this attribute is deprecated, it has been used in several of the examples in this chapter so that you can clearly see where the edge of each table cell is.

### ***The cellpadding Attribute (Deprecated)***

The `cellpadding` attribute is used to create a gap between the edges of a cell and its contents. The value for this attribute determines the amount of space or padding inside each wall of the cell, specified either in pixels or as a percentage value (where the percentage is a percentage of the width of each cell).

```
cellpadding="5" or cellpadding="2%"
```

As you can imagine, if two table cells sat next to each other and both contained writing, there could be a problem. If there were not a gap between the edge of the cells and the writing, the words would butt up against each other, making them hard to read. Similarly, if there were a border around each cell and the text touched the border, it could be hard to read. Therefore, adding padding to cells makes their contents easier to read.

# Chapter 4: Tables

---

This attribute has been replaced by the `padding` property in CSS.

## The `cellspacing` Attribute (Deprecated)

The `cellspacing` attribute is used to create a space between the borders of each cell. The value for this attribute can be either the amount of space you want to create between the cells, in pixels, or a percentage value.

```
cellspacing="6" or cellspacing="2%"
```

This attribute has been replaced by the `margin` property in CSS.

## The `dir` Attribute

The `dir` attribute is supposed to indicate the direction of text that is used in the table. Possible values are `ltr` for left to right text and `rtl` for right to left (for languages such as Hebrew and Arabic):

```
dir="rtl"
```

If you use the `dir` attribute with a value of `rtl` on the `<table>` element, then the cells appear from the right first, and each consecutive cell is placed to the left of that one.

## The `frame` Attribute (Deprecated)

The `frame` attribute is supposed to control the appearance of the outermost border of the whole table, referred to here as its *frame*, with greater control than the `border` attribute. If both the `frame` and `border` attributes are used, the `frame` attribute takes precedence. The syntax is:

```
frame="frameType"
```

The following table shows the possible values for the `frame` attribute.

| Value  | Purpose                                       |
|--------|---|
| void   | No outer border (the default)                 |
| above  | A border on the top only                      |
| below  | A border on the bottom only                   |
| hsides | A border on the top and bottom                |
| lhs    | A border on the left side of table            |
| rhs    | A border on the right side of table           |
| vsides | A border on the left and right sides of table |
| box    | A border on all sides                         |
| border | A border on all sides                         |

Support for the `frame` attribute is not perfect across browsers, and its role has been replaced by the CSS `border` property, which can give better results.

### ***The rules Attribute (Deprecated)***

The `rules` attribute is used to indicate which inner borders of the table should be displayed. For example, you can just specify that the rows *or* columns should have lines between each of them. Here is the syntax; the default value is `none`.

```
rules="ruleType"
```

The following table shows the possible values for the `rules` attribute:

| Value               | Purpose   |
|---------------------|---|
| <code>none</code>   | No inner borders (the default)  |
| <code>groups</code> | Displays inner borders between all table groups (groups are created by the <code>&lt;thead&gt;</code> , <code>&lt;tbody&gt;</code> , <code>&lt;tfoot&gt;</code> , and <code>&lt;colgroup&gt;</code> elements) |
| <code>rows</code>   | Displays horizontal borders between each row  |
| <code>cols</code>   | Displays vertical borders between each column   |
| <code>all</code>    | Displays horizontal and vertical borders between each row and column  |

Again, support for this attribute is not perfect, and it has been deprecated in favor of the CSS `border` property, which achieves better results.

### ***The summary Attribute***

The `summary` attribute is supposed to provide a summary of the table's purpose and structure for non-visual browsers such as speech browsers or Braille browsers. The value of this attribute is not rendered in IE or Firefox, but if the text before the table doesn't clearly explain what will be found in the table, you should aim to use it in your pages:

```
summary="Table shows the operating profit for the last four quarters. The first column indicates the quarter, the second indicates outgoings, the third indicates receipts, and the fourth indicates profit."
```

### ***The width Attribute (Deprecated)***

The `width` attribute is used to specify the width of the table, and usually its value is given in pixels.

```
width="500"
```

It is also possible to use a percentage of the available space, the available space generally being the width of the browser, unless the table sits inside another element within the body of the page, in which case it is a percentage of the size of the containing element.

```
width="90%"
```

### ***The <tr> Element Contains Table Rows***

The `<tr>` element is used to contain each row in a table. Anything appearing within a `<tr>` element should appear on the same row. It can carry five attributes, three of which have been deprecated in favor of using CSS.

```
align bgcolor char charoff valign
```

### ***The align Attribute (Deprecated)***

The `align` attribute specifies the position of the content of all of the cells in the row.

```
align="alignment"
```

The table that follows lists the possible values for the `align` attribute:

| Value   | Purpose  |
|---------|--|
| left    | Content is left-aligned.   |
| right   | Content is right-aligned.  |
| center  | Content is centered horizontally within the cell.  |
| justify | Text within the cell is justified to fill the cell.  |
| char    | Cell contents are aligned horizontally around the first instance of a specific character (for example, numbers could be aligned around the first instance of a decimal point). |

By default, any `<td>` cells are usually left-aligned, whereas any `<th>` cells are usually centered. While Firefox 2+ supports justified text, IE does not, and neither IE nor Firefox support the value `char`.

### ***The bgcolor Attribute (Deprecated)***

The `bgcolor` attribute sets the background color for the row. The value of this attribute should be either a color name or a hex code or color value, as discussed in Appendix D.

```
bgcolor="red"
```

The `bgcolor` attribute is commonly used on the `<tr>` element to shade alternate rows of a table with different colors, thus making it easier to read across each row.

### ***The char Attribute***

If the `align` attribute has been given a value of `char`, then the `char` attribute is used to specify that the contents of each cell within the row will be aligned around the first instance of a particular character

known as an *axis character*. The default character for this attribute is the decimal place, and the idea is that decimal figures would be aligned by the decimal point like so:

```
13412.22
 232.147
2449.6331
  2.12
```

The syntax is as follows:

```
char="."
```

Unfortunately, this potentially very helpful attribute is not supported at the time of this writing, and there is no requirement for browsers to support it.

### The *charoff* Attribute

The `charoff` attribute is used in association with the `char` attribute, and indicates an offset for the `char` attribute. For example, if it is given a value of 2, then the elements are aligned with the character that is two characters along from the one specified by the `char` attribute. It can also take a negative value.

If this attribute is omitted, the default behavior is to make the offset the equivalent of the longest amount of text content that appeared before the character specified in the `char` attribute.

```
charoff="2"
```

Unfortunately, this attribute is not supported at the time of this writing, and there is no requirement for browsers to support it.

### The *valign* Attribute (Deprecated)

The `valign` attribute specifies the vertical alignment of the contents of each cell in the row. The syntax is as follows:

```
valign="verticalPosition"
```

The table that follows shows the possible values of the `valign` attribute:

| Value    | Purpose   |
|----------|---|
| top      | Aligns content with the top of the cell   |
| middle   | (Vertically) aligns content in the center of a cell   |
| bottom   | Aligns content with the bottom of the cell  |
| baseline | Aligns content so that the first line of text in each cell starts on the same horizontal line |

### ***The <td> and <th> Elements Represent Table Cells***

Every cell in a table will be represented by either a `<td>` element for cells containing table data or a `<th>` element for cells containing table headings.

By default, the contents of a `<th>` element are usually displayed in a bold font, horizontally aligned in the center of the cell. The content of a `<td>` element, meanwhile, will usually be displayed left-aligned and not in bold (unless otherwise indicated by CSS or another element).

The `<td>` and `<th>` elements can both carry the same set of attributes, and the attribute only applies to that one cell carrying it. Any effect these attributes have will override settings for the table as a whole or any containing element (such as a row).

In addition to the universal attributes and the basic event attributes, the `<td>` and `<th>` elements can also carry the following attributes:

```
abbr align axis bgcolor char charoff colspan headers
height nowrap rowspan scope valign width
```

### ***The abbr Attribute***

The `abbr` attribute is used to provide an abbreviated version of the cell's content.

```
abbr="description of services"
```

While the major browsers do not currently support this attribute, it's possible that it will be used by some of the increasing number of devices with small screens accessing the Internet. For example, if a browser with a small screen is being used to view the page, the content of this attribute could be displayed instead of the full content of the cell.

### ***The align Attribute (Deprecated)***

The `align` attribute sets the horizontal alignment for the content of the cell.

```
align="alignment"
```

The possible values for the `align` attribute are `left`, `right`, `center`, `justify`, and `char`, each of which was described earlier in “The `<tr>` Element Contains Table Rows” section.

### ***The axis Attribute***

The `axis` attribute allows you to add conceptual categories to cells and therefore represent *n*-dimensional data. The value of this attribute would be a comma-separated list of names for each category the cell belonged to.

```
axis="heavy, old, valuable"
```

Rather than having a visual formatting effect, this attribute allows you to preserve data, which then may be used programmatically, such as querying for all cells belonging to a certain category.



### ***The bgcolor Attribute (Deprecated)***

The `bgcolor` attribute sets the background color for the cell. The value of this attribute should be either a color name or a hex code — both are covered in Appendix D.

```
bgcolor="red"
```

It has been replaced by the `background-color` property in CSS.

### ***The char Attribute***

The `char` attribute specifies a character, the first instance of which should be used to horizontally align the contents of a cell. (See the full description in the “The `char` Attribute” subsection within the “The `<tr>` Element Contains Table Rows” section earlier in the chapter.)

### ***The charoff Attribute***

The `charoff` attribute specifies the number of offset characters that can be displayed before the character specified as the value of the `char` attribute. (See the full description in the “The `charoff` Attribute” subsection within the “The `<tr>` Element Contains Table Rows” section earlier in the chapter.)

### ***The colspan Attribute***

The `colspan` attribute is used when a cell should span across more than one column. The value of the attribute specifies how many columns of the table a cell will span across. (See the section “Spanning Columns Using the `colspan` Attribute” later in this chapter.)

```
colspan="2"
```

### ***The headers Attribute***

The `headers` attribute is used to indicate which headers correspond to that cell. The value of the attribute is a space-separated list of the header cells’ `id` attribute values:

```
headers="income q1"
```

The main purpose of this attribute is to support voice browsers. When a table is being read to you, it can be hard to keep track of which row and column you are on; therefore, the `header` attribute is used to remind users which row and column the current cell’s data belongs to.

### ***The height Attribute (Deprecated)***

The `height` attribute allows you to specify the height of a cell in pixels, or as a percentage of the available space:

```
height="20" or height="10%"
```

It has been replaced by the CSS `height` property.

### ***The nowrap Attribute (Deprecated)***

The `nowrap` attribute is used to stop text from wrapping onto a new line within a cell. You would use `nowrap` only when the text really would not make sense if it were allowed to wrap onto the next line (for example, a line of code that would not work if it were spread across two lines). When it was initially

# Chapter 4: Tables

introduced in HTML, it was used without an attribute value, but that would not be allowed in XHTML. Rather, you would have to use the following:

```
nowrap="nowrap"
```

## The rowspan Attribute

The `rowspan` attribute specifies the number of rows of the table a cell will span across, the value of the attribute being the number of rows the cell stretches across. (See the example in the section “Spanning Rows Using the `rowspan` Attribute” later in this chapter.)

```
rowspan="2"
```

## The scope Attribute

The `scope` attribute can be used to indicate which cells the current header provides a label or header information for. It can be used instead of the `headers` attribute in basic tables, but does not have much support.

```
scope="range"
```

The table that follows shows the possible values of the attribute:

| Value    | Purpose  |
|----------|--|
| row      | Cell contains header information for that row.   |
| col      | Cell contains header information for that column.  |
| rowgroup | Cell contains header information for that rowgroup (a group of cells in a row created using the <code>&lt;thead&gt;</code> , <code>&lt;tbody&gt;</code> , or <code>&lt;tfoot&gt;</code> elements).             |
| colgroup | Cell contains header information for that colgroup (a group of columns created using the <code>&lt;col&gt;</code> or <code>&lt;colgroup&gt;</code> element, both of which are discussed later in the chapter). |

## The valign Attribute (Deprecated)

The `valign` attribute allows you to specify the vertical alignment for the content of the cell. Possible values are `top`, `middle`, `bottom`, and `baseline`, each of which was discussed earlier in the chapter in the subsection entitled “The `valign` Attribute” within the section “The `<tr>` Element Contains Table Rows.”

## The width Attribute (Deprecated)

The `width` attribute allows you to specify the width of a cell in pixels, or as a percentage of the table:

```
width="150" or width="30%"
```

You only need to specify the `width` attribute for the cells in the first row of a table, and the rest of the rows will follow the first row's cell widths.

If you had already specified a `width` attribute for the `<table>` element, and the widths of individual cells add up to more than that width, most browsers will squash those cells to fit them into the width of the table.

You can also add a special value of `*`, which means that this cell will take up the remaining space available in the table. So if you have a table that is 300 pixels wide and the first two cells in a row are specified as being 50 pixels wide, if the third cell has a value of `*`, it will take up 200 pixels — the remaining width of the table. If the width of the table had not been specified, then the third column would take up the remaining width of the browser window.

It is worth noting that you cannot specify different widths for different `<td>` elements that belong in the same column. So, if the first row of a table had three `<td>` elements whose widths were 100 pixels, the second row could not have one `<td>` element whose width was 200 pixels and two that were 50 pixels.

## Try It Out      An Opening Hours Table

In this example, you will create a table that shows the opening hours of the Example Café web site we have been working on throughout the book. The table will look like the one shown in Figure 4-9.



The screenshot shows a Mozilla Firefox browser window titled "Example Cafe - opening hours and how to find Example Cafe in Newquay, Cornwall". The page content includes the "example cafe" logo, navigation links (HOME MENU RECIPES OPENING CONTACT), the heading "Opening hours and serving times", and a text line "We are open from 7am - 3.30pm Monday to Friday and 8am - 4pm Saturday and Sunday". Below this is a table with 8 columns (days of the week) and 3 rows (Breakfast, Lunch, and an empty row for the afternoon).

|           | Monday           | Tuesday          | Wednesday        | Thursday         | Friday           | Saturday         | Sunday           |
|-----------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| Breakfast | 7:00am - 10:00am | 7:00am - 10:00am | 7:00am - 10:00am | 7:00am - 10:00am | 7:00am - 11:00am | 8:00am - 11:30pm | 8:00am - 11:30pm |
| Lunch     | 11:30am - 2:30pm | 11:30am - 2:30pm | 11:30am - 2:30pm | 11:30am - 2:30pm | 11:30am - 2:30pm | 11:30am - 3:30pm | 11:30am - 3:30pm |
|           |                  |                  |                  |                  |                  |                  |                  |

Figure 4-9

1. Start off by opening the `contact.html` file in your text or XHTML editor; we will be adding a table to show serving hours beneath the e-mail link.
2. The table is contained within the `<table>` element and its content is then written out a row at a time. The table has three rows and eight columns.

## Chapter 4: Tables

---

Starting with the top row, you have eight table heading elements. The first `<th>` element is empty because the top-left corner cell of the table is empty. The next seven elements contain the days of the week.

In the second row of the table, the first cell acts as a heading for that row, indicating the meal (breakfast). The remaining cells show what times these meals are served. The third row follows the same format as the second row but shows times for lunch.

```
<table>
  <tr>
    <th></th>
    <th>Monday</th>
    <th>Tuesday</th>
    <th>Wednesday</th>
    <th>Thursday</th>
    <th>Friday</th>
    <th>Saturday</th>
    <th>Sunday</th>
  </tr>
  <tr>
    <th>Breakfast</th>
    <td>7:00am - 10:00am</td>
    <td>7:00am - 10:00am</td>
    <td>7:00am - 10:00am</td>
    <td>7:00am - 10:00am</td>
    <td>7:00am - 11:00am</td>
    <td>8:00am - 11:30pm</td>
    <td>8:00am - 11:30pm</td>
  </tr>
  <tr>
    <th>Lunch</th>
    <td>11:30am - 2:30pm</td>
    <td>11:30am - 2:30pm</td>
    <td>11:30am - 2:30pm</td>
    <td>11:30am - 2:30pm</td>
    <td>11:30am - 3:30pm</td>
    <td>11:30am - 3:30pm</td>
  </tr>
</table>
</body>
```

As long as you accept that each row is written out in turn, you will have no problem creating quite complex tables.

3. Save your file as `contact.html`.
-