30

CHAPTER

# Formatting Tables

In this chapter, you learn about using CSS to format tables. As you read in Chapter 20, CSS1 does not provide as many ways to affect formatting of tables as it does formatting of other elements. Fortunately, the W3C has remedied this problem with CSS2. As always, the transition isn't seamless, because not all browsers fully support CSS2. This chapter incorporates both CSS1 and CSS2 to show you how to format every aspect of your HTML table.

## Controlling Table Alignment

CSS provides the align property to enable you to specify table alignment. The align property for tables works exactly as it does for aligning paragraphs or any other block-level element. You can choose to align the table on the left, in the center, on the right, or fully justified across the canvas. Look at the code for an example. You can specify the align property for the `TABLE` selector, for a class of the `TABLE` selector, or for an ID that affects the `TABLE` element.

```
TABLE {
        text-align: center;
}
TABLE.right-leaning {
        text-align: right;
}
TABLE#full {
        text-align: justify;
}
```

## Setting Horizontal Cell Alignment

You can specify horizontal cell alignment in four places: at the cell level, at the row level, at the column level, or at the row group level.

### At the cell level

Specifying alignment at the cell level is easy. Include the align attribute in the TD element. The problem with this is you could end up defining this attribute in every single cell. If you can define this attribute at a higher level, this can save you a lot of typing and reduce the chance of introducing errors into your TABLE definition.

```
<TABLE>
  <TR>
        <TH align="center">Vacation Location
        <TH align="center">Avg. Temp.
        <TH align="center">Package Deal
  <TR>
        <TD align="left">Ireland
        <TD align="char" char="&deg;">68&deg;
        <TD align="char" char=".">$799.95
  <TR>
        <TD align="left">Greek Islands
        <TD align="char" char="&deg;">84&deg;
        <TD align="char" char=".">$649.95
  <TR>
        <TD align="left">Cancun
        <TD align="char" char="&deg;">85&deg;
        <TD align="char" char=".">$729.99
</TABLE>
```

Using CSS, you can move some of this formatting into classes. There is no character alignment in CSS, so some formatting would have to remain in the HTML. The style sheet would look like this:

```
TD.left {
        align: left;
}
TH.center {
        align: center;
}
```

And the HTML would look like this:

```
<TABLE>
  <TR>
        <TH class="center">Vacation Location
        <TH class="center">Avg. Temp.
        <TH class="center">Package Deal
  <TR>
        <TD class="left">Ireland
        <TD align="char" char="&deg;">68&deg;
        <TD align="char" char=".">$799.95
  <TR>
        <TD class="left">Greek Islands
        <TD align="char" char="&deg;">84&deg;
        <TD align="char" char=".">$649.95
  <TR>
        <TD class="left">Cancun
        <TD align="char" char="&deg;">85&deg;
        <TD align="char" char=".">$729.99
</TABLE>
```

### At the row level

As you can see in the previous example, defining alignment at the cell level is impractical — especially for the first row, where every cell has the same class. Moving alignment to the row level makes more sense. The style sheet would now look like this:

```
TD.left {
        text-align: left;
}
TR.center TH.center {
        text-align: center;
}
```

And the HTML would look like this:

```
<TABLE>
  <TR class="center">
        <TH>Vacation Location
        <TH>Avg. Temp.
        <TH>Package Deal
  <TR>
        <TD align="left">Ireland
        <TD align="char" char="&deg;">68&deg;
        <TD align="char" char=".">$799.95
  <TR>
        <TD align="left">Greek Islands
        <TD align="char" char="&deg;">84&deg;
        <TD align="char" char=".">$649.95
  <TR>
        <TD align="left">Cancun
        <TD align="char" char="&deg;">85&deg;
        <TD align="char" char=".">$729.99
</TABLE>
```

Notice the alignment for the first row was moved from the cell level to the row level. Already, the HTML looks better.

## At the column level

Unfortunately, none of the other rows lend themselves to moving the alignment attribute to the row level. But all of them share column formatting. They all have a left column with left alignment; a center column with character alignment that lines up on the degree sign (°), defined using the `&deg;` entity; and a right column with character alignment that lines up on the decimal point.

To take advantage of column-level alignment, you have to define columns. The previous example doesn't define columns, but the following example does. All the alignment formatting for rows after the first row is moved to the column level. The style sheet needs to be modified to look like this:

```
COL.left TD.left {
        text-align: left;
}
TR.center TH.center {
        text-align: center;
}
```

Because there is no character alignment in CSS, the formatting for the second and third columns would still need to be specified in the HTML.

```
<TABLE>
  <COL class="left">
  <COL align="char" char="&deg;">
  <COL align="char" char=".">
  <TR align="center">
        <TH>Vacation Location
        <TH>Avg. Temp.
        <TH>Package Deal
  <TR>
        <TD>Ireland
        <TD>68&deg;
        <TD>$799.95
  <TR>
        <TD>Greek Islands
        <TD>84&deg;
        <TD>$649.95
  <TR>
        <TD>Cancun
        <TD>85&deg;
        <TD>$729.99
</TABLE>
```

Even though the first row would be affected by the alignment formatting defined at the column level, the row-level alignment overrides the column-level formatting, so all the cells in the first row will have center alignment. Finally, the HTML looks like something you probably wouldn't mind maintaining.

### At the row group level or column group level

The previous example doesn't lend itself to the use of the `THEAD`, `TFOOT`, `TBODY`, or `COLGROUP` elements to specify alignment, but you can specify alignment at either the row group level or the column group level.

```
COLGROUP.center {
        text-align: center;
}
TBODY.right {
        text-align: right;
}
```

## Setting Vertical Cell Alignment

New to CSS2 is the `vertical-align` property. You can specify vertical alignment (called *valign* in HTML 3.2) in your style sheet to apply to `TD` elements. See the code example for how to implement the `TD` elements.

```
TD.mid {
        vertical-align: middle;
}
```

You may use a keyword or specify a percentage value. The keywords are as follows:

- **top.** Sets the top of the cell to align with the top of the row in which the cell is defined.
- **middle.** Sets the middle of the cell to align with the middle of the row in which the cell is defined.
- **bottom.** Sets the bottom of the cell to align with the bottom of the row in which the cell is defined.
- **baseline.** This is a relative specification. The baseline of the row is the baseline of the top line of the text in the row. The baseline of the top line of text in the row depends on the font sizes used in each of the cells. The browser determines the baseline of the first row of text and aligns the baseline of the text in this cell with the baseline of the row.
- **sub.** This subscripts the cell.
- **super.** This superscripts the cell.

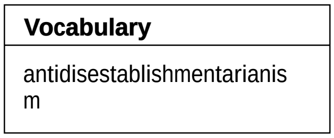
- ✦ **text-top.** Aligns the top of the cell with the top of the font of the highest text in that row.
- ✦ **text-bottom.** Aligns the bottom of the cell with the bottom of the font of the lowest text in that row.

## Specifying Table and Cell Widths

You can specify widths for both the table and individual cells. With cell specifications, specify width at the column level or at the cell level. For both cells and tables, you can either specify widths as absolute values or as relative values.

### Absolute values

Before you decide to specify your table width using absolute values, be sure you know how each cell will render. Otherwise, you could end up with cells that aren't wide enough to hold an entire word. Tables don't do a good job of breaking words at the syllable or hyphenating your words when you specify too small a width for cells. In fact, if you specify an absolute value for cell height and an absolute value for cell width, your cell might be too small to display all your text. In this case, the text might spill over into the next cell, crossing rules!



| Vocabulary |
|---|
| antidisestablishmentarianis m |

When you specify absolute widths, you are specifying them in pixels. A screen can be 640 pixels wide; 800 pixels wide; or 1,024 pixels wide (even more for super-high-resolution monitors), but the browser will not always be set to fill the entire screen. You have no way of predicting how wide the browser window will be. The attribute used for specifying width is the width attribute.

```
<TABLE width="200">
  <COL width="100">
  <COL width="50" span="2">
  ... rest of table...
</TABLE>
```

In the previous example, the table is set to 200 pixels wide, the first column is 100 pixels wide, and the second and third columns are each 50 pixels wide. Normally you wouldn't specify both `TABLE` width and `COL` width absolutely.

## Relative values

Whenever possible, you want to use relative values to specify table width. You can specify the table width as a percentage of the screen.

```
<TABLE width="40%">
  ... rest of table...
</TABLE>
```

You can also specify relative cell widths within a table for which you have specified an absolute width. The example in the previous section about absolute widths would probably be better defined as follows:

```
<TABLE width="200">
  <COL width="50%">
  <COL width="25%" span="2">
  ... rest of table...
</TABLE>
```

Another way to define cell widths is to use the `0*` value. The `0*` value is the way you instruct the browser to use the minimum width necessary to construct a table.

```
<TABLE>
  <COL width="0*">
  <COL width="100" span="2">
  ... rest of table...
</TABLE>
```

## Specifying width in style sheets

You can also use style sheets to specify widths. The property is called `width`. The last two examples could be reworked, using style sheets, into the following code:

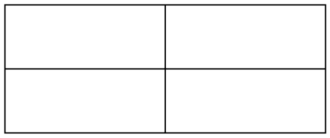
```
<TABLE class="this-one">
  <COL ID="half">
  <COL ID="quarter" span="2">
  ... rest of table...
</TABLE>
<TABLE>
  <COL class="min">
  <COL class="this-one" span="2">
  ... rest of table...
</TABLE>
```

The style sheet for these two tables would look like this:

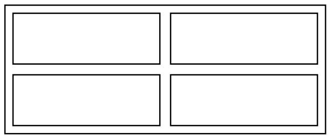
```
TABLE.this-one {
        width: 200;
}
#half {
        width: 50%;
}
#quarter {
        width: 25%;
}
COL.min {
        width: 0*;
}
COL.this-one {
        width: 100;
}
```

## Adding Cell Spacing

You can specify cell spacing within a table. The property is called *cell spacing* (cell spacing as an attribute of the `TABLE` element). *Cell spacing* is the space between the border on the inside of one cell and the border on the inside of the next cell or the outside border of the table. When you define borders, borders are actually around each cell, and a separate border is around the entire table. If the borders are thin and there is no cell spacing, it will look like it is one line that surrounds the cells and attaches to the border around the table. The more cell spacing you define, the clearer it will be that these are separate lines.



The previous table has no cell spacing.



The previous table has cell spacing. The spacing attribute is defined as part of the `TABLE` element as follows:

```
<TABLE cellspacing="3">
```

Or, as part of a style sheet:

```
TABLE {
        cellspacing: 3
}
```

New to CSS2, you can specify different values for horizontal space between cells and for vertical space between cells. The first value supplied to the `cellspacing` property is for horizontal cell spacing; the second is for vertical cell spacing. If you provide only one, it applies to both kinds of cell spacing.

Also new to CSS2, you can collapse borders, removing the cell spacing altogether. This makes your table look more like the way a table appears in a word-processing document. The property is called `border-collapse`. This applies only to table elements: `TABLE`, `COLGROUP`, `COL`, `THEAD`, `TFOOT`, `TBODY`, `TR`, `TH`, and `TD`.

```
TABLE {
        border-collapse: collapse;
}
```

The default value of `border-collapse` is *separate*. You can choose between the keywords *collapse* and *separate*.

## Defining Cell Padding

Cell padding is the white space between the contents of a cell and the borders of a cell. Cell padding can be defined at the `TABLE` level, the column group or column level, the row group or row level, or the cell level. By default, browsers render tables without any cell padding. This tends to make the widest element in each cell look crowded.

The property is the same `padding` property used to define padding for any block-level element that takes advantage of the box formatting model.

```
TABLE {
        padding: 1em;
}
```

Even if you specify an absolute width for a table or a cell, the `padding` attribute has an effect. If you specify an absolute width and padding, the browser will make the width of the cell in which the contents appear smaller, allowing room for the padding.

## Using Colors in Tables

Within a table, you can assign color to several elements. These elements include the borders, the rules, the background color, and the text color. You can specify all these things as the same for a table or you can specify different colors for every rule, border, cell text color, and cell background color. The properties are the same as for the rest of CSS: `color`, `background-color`, and the variations on the `border`/`border-color` property.

```
COLGROUP {
        border: thin solid black;
        background-color: white;
        color: black;
}
COLGROUP.highlight {
        border-left: thick solid blue;
        background-color: yellow;
        color: navy;
}
```

## Defining Rules and Borders

By default, tables in HTML have a border-width set to zero. This means all rules and borders are turned off. If you take the small action of setting border-width to 1, you will suddenly have both borders and rules.

*Rules* are the lines between cells. The capability to define these independently of the border, which is the line around the table, is new to HTML 4. The three attributes for defining borders are: `rules`, `frame`, and `border`. The border attribute defines the width of the frame and rules; the value must be defined as a valid measurement using the units listed in Chapter 29.

The `frame` attribute takes a keyword value; valid values are as follows:

- **void.** No borders are displayed on the table; this is the default value.
- **above.** Only a top border is displayed.
- **below.** Only the bottom border is displayed.
- **hsides.** The top and bottom borders are displayed.
- **vsides.** The right and left borders are displayed.
- **lhs.** Only the left-hand side border is displayed.
- **rhs.** Only the right-hand side border is displayed.
- **box.** All four sides are displayed.
- **border.** All four sides are displayed.

The `rules` attribute also takes a keyword value. The `rules` attribute specifies internal lines. Valid values for the `rules` attribute are as follows:

- ✦ **none.** No rules are displayed. This is the default value.
- ✦ **group.** Rules are displayed only between groups.
- ✦ **rows.** Rules are displayed between rows.
- ✦ **cols.** Rules are displayed between columns.
- ✦ **all.** Rules are displayed between all cells.

## From Here

Cross-Reference

Jump to Chapter 33 and learn about CSS absolute-positioning tricks.

Proceed to Chapter 31 to learn about using CSS to define fonts.

## Summary

Using a combination of CSS1, CSS2, and HTML 4, you can define the formatting of your tables. Controlling table alignment, horizontal alignment of cell contents in your cells, and vertical alignment of cell contents in your rows is relatively straightforward. In this chapter, you learned about specifying table and cell widths as both relative and absolute values. You also learned about cell spacing, cell padding, defining rules and borders, and defining colors in tables.

✦ ✦ ✦