CHAPTER

13

Building Float-Based Layouts

Float-based layouts take advantage of the `float` property to position elements side by side and create columns on a web page. As described in Chapter 7 (page 220), you can use this property to create a wrap-around effect for, say, a photograph, but when you apply it to a `<div>` tag, `float` becomes a powerful page-layout tool. The `float` property moves a page element to one side of the page (or other containing block). Any HTML that appears below the floated element moves up on the page and wraps around the float.

The `float` property accepts one of three different values—`left`, `right`, and `none`. To move an image to the right side of the page, you could create this class style and apply it to the `<img>` tag:

```
.floatRight { float: right; }
```

The same property applied to a `<div>` tag full of content can also create a sidebar:

```
.sidebar {
  float: left;
  width: 170px;
}
```

Figure 13-1 shows these two styles in action.

NOTE The none value turns off any floating and positions the element like a normal, unfloated element. It's useful only for overriding a float that's already applied to an element. You may have an element with a particular class such as `.sidebar` applied to it, with that element floating to the right. But on one page you may want an element with that class to *not* float, but to be placed within the flow of the page, like this Note box. By creating a more specific CSS selector (see page 480) with `float: none`, you can prevent that element from floating.

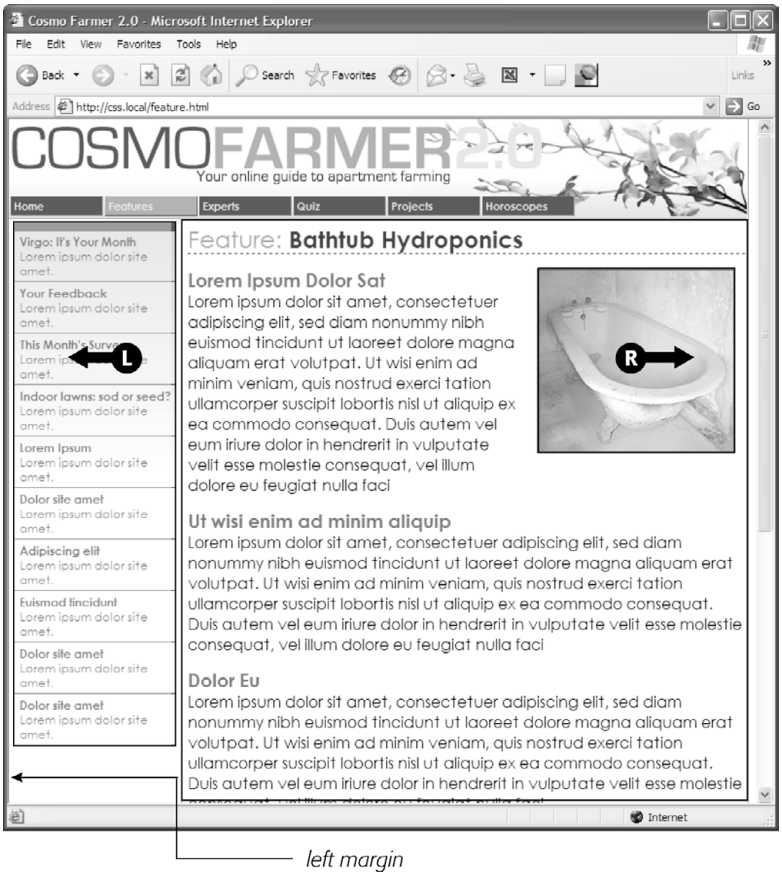

FIGURE 13-1

You can use the `float` property to lay out a web page with multiple columns. On this page, a block of content is floated to the left edge. The sidebar has a set width, but the main content doesn't, which makes this design a liquid layout. The main section of the page simply expands to fill the width of the browser window. In the upper right, the bathtub photo is floated to the right.

A simple two-column design like Figure 13-1 requires just a few steps:

1. **Wrap each column in a `<div>` tag with an ID or class attribute.**

   In Figure 13-1, the news items listed in the left sidebar are wrapped in one `<div>`—`<div class="news">`—and the main content in another div—`<div class="main">`.

2. **Float the sidebar `<div>` either right or left.**

   When you work with floats, the source order (the order in which you add HTML to a file) is important. The HTML for the floated element must appear *before* the HTML for the element that wraps around it.

Figure 13-2 shows three two-column layouts. The diagrams on the left side show the page's HTML source order: A `<div>` for the banner, followed by a `<div>` for the sidebar, and, lastly, a `<div>` for the main content. On the right side, you see the actual page layout. The sidebar comes *before* the main content in the HTML so it can float either left (top, bottom) or right (middle).

3. **Set a width for the floated sidebar.**

   Unless you're floating an image with a predefined width, you should always give your floats a width. This way, you create a set size for the floated element, letting the browser make room for other content to wrap into position.

   The width could be a fixed size like `170px` or `10em`. You can also use percentages for a flexible design that's based on the width of the browser window. (See page 158 for more about the pros and cons of the different measurement units.) If the sidebar is 20 percent wide, and the browser window is 700 pixels wide, then the sidebar will be 140 pixels wide. But if your visitor resizes the window to 1000 pixels, then the sidebar grows to 200 pixels. Fixed-width sidebars are easier to design for, since you don't have to consider all the different widths the sidebar might stretch to.

   However, percentages let you maintain the same proportions between the two columns, which can be more visually pleasing. In addition, percentages make your designs more flexible, since the overall proportion of the page can adjust to fit the screen size, something that's important when creating responsive web designs, which you'll learn about in the next chapter.

NOTE When the overall page design is a fixed width (as described on page 406), percentage width values for the sidebar are based on the fixed-width containing element. The width isn't based on the window size and won't change when the browser window changes size.

4. **Add a left margin to the main content.**

   If the sidebar is shorter than the other content on the page, the text from the main column wraps underneath the sidebar, ruining the look of two side-by-side columns (see Figure 13-12 for an example). Adding a left margin that's equal to or greater than the width of the sidebar indents the main content of the page, creating the illusion of a second column:

   ```
   .main { margin-left: 180px; }
   ```

   By the way, it's usually a good idea to make the left margin a little bigger than the width of the sidebar: This creates some empty space—a white gutter—between the two elements. So, when you use percentages to set the width of the sidebar, use a slightly larger percentage value for the left margin.

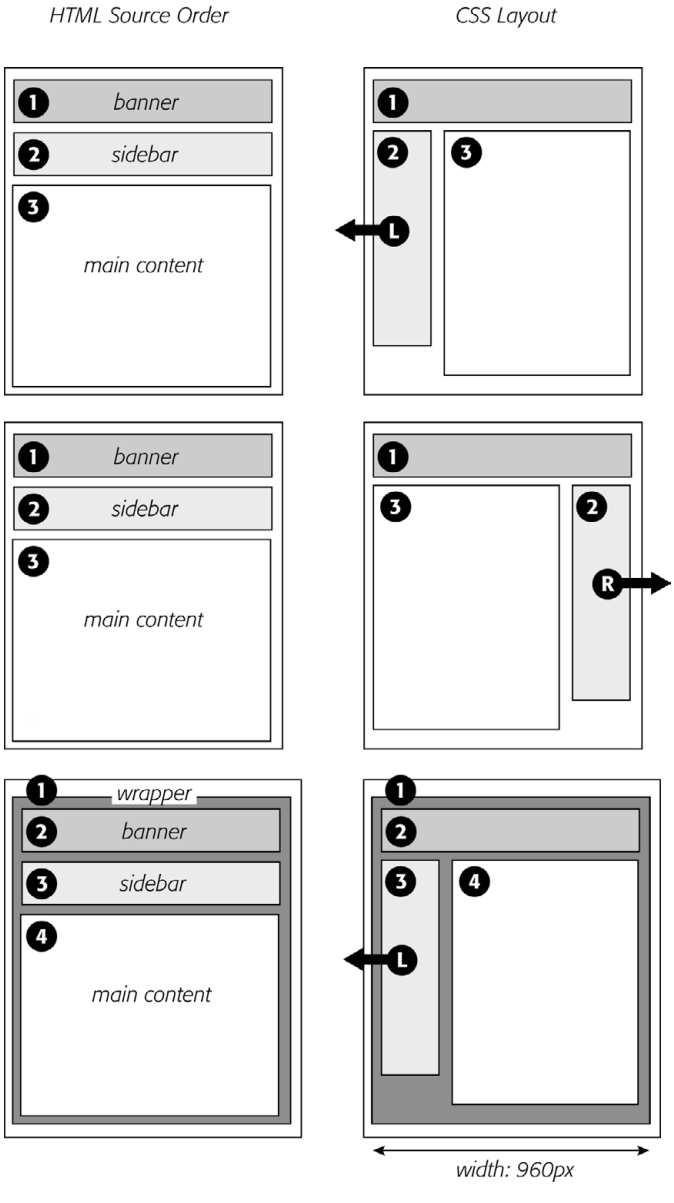

FIGURE 13-2

Creating a two-column layout is a simple matter of floating a `<div>` *tag to the left (top). To make a sidebar move from the left to right side of the page (middle), just change the sidebar's CSS styling to* `float: right`. *You don't need to make any other changes to your CSS or HTML.*

Avoid setting a width for the main content div. It's not necessary, since browsers simply expand it to fit the available space. Even if you want a fixed-width design, you don't need to set a width for the main content div, as you'll see in the next section.

Applying Floats to Your Layouts

Now that you've learned a basic two-column liquid layout, you can adapt it in countless ways. Converting it into a fixed-width layout is a snap. Simply wrap all the tags within the page's body inside *another* `<div>` (like `<div class="wrapper">`). Then, create a style for that new container element that has a set width, such as 960 pixels (see Figure 13-2, bottom). That width setting constrains everything inside the container box.

TIP It's also possible to create a fixed-width page without resorting to the extra wrapper div: set a width on the `<body>` tag. You already saw an example of this technique in the tutorial on page 225.

Expanding it into a three-column design isn't difficult, either (Figure 13-3). First, add another `<div>` between the two columns and float it to the right. Then add a right margin to the middle column, so that if the text in the middle column runs longer than the new right sidebar, it won't wrap underneath the sidebar.

The rest of this section explores more CSS layout techniques that use float-based layouts.

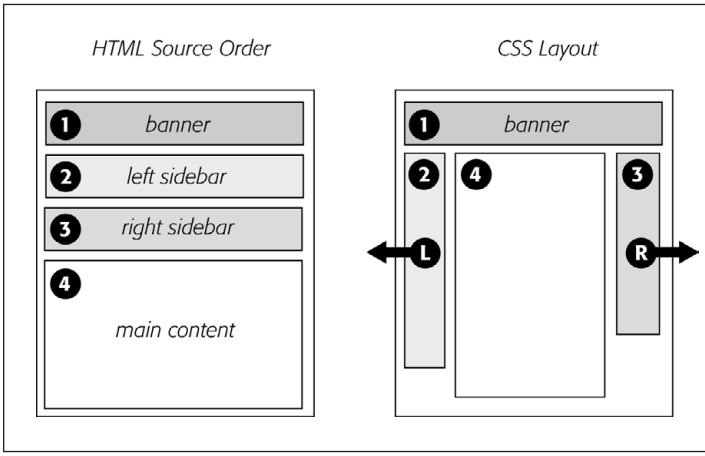

FIGURE 13-3
A three-column design uses the same concepts used to create a two-column design. In this case, you float both the left and right sidebars and add both left and right margins to the center column. The lefthand diagram shows the order of the HTML; the right side shows what the web page looks like.

Floating All Columns

It's perfectly possible to float *every* column, not just the left and right sidebars. You could float the first sidebar to the left, the middle column to the left, and the right sidebar to the right, as shown in Figure 13-4, top. This approach lets you put more

than three columns in your design. You can float four or more columns, as long as there's room for all the floats to fit side by side.

When you float all columns in a design, you need to pay close attention to the widths of each column. If the total width of all the columns is more than the space available—for example, if the browser window is smaller or the columns are placed inside another `<div>` with a set width—then the last column drops down below the others. (You can read a solution to this dropping float problem on page 480.)

UP TO SPEED

You Don't Have to Reinvent the Wheel

If terms like *liquid layout* and *containing element* sound a little intimidating, don't give up. First of all, the tutorials beginning on page 440 walk you step by step through the process of laying out web pages with CSS. But there's no law saying you have to create your own CSS layouts from scratch. On the Web, you'll find plenty of pre-built and tested designs you can make your own. The LayoutGala site offers 40 different CSS designs that work in all common browsers (*http://blog.html.it/layoutgala/*). The designs are just basic skeletons consisting of `<div>` tags and the CSS that positions them. All you need to do is fill them with your own design touches like font styling and imagery.

There are also quite a few *layout generators*—online tools that let you customize basic requirements like the number of columns you want, whether you're after a liquid or fixed layout, and so on. The Gridinator (*http://gridinator.com*) provides a simple tool for creating a complex multicolumn grid system (see the box on page 430). You can then download HTML and CSS files with the code created for you.

In addition, floating more than just the sidebars lets you change the order of your divs in the HTML. Take, for example, the left diagram in Figure 13-3, which shows the order of the `<div>` tags for that page. Because of the way floated elements work, they must appear before any content that wraps around them, so in this example, the main content area must go *after* the sidebars.

The order of the `<div>` tags in the HTML may not seem like a big deal until you try to browse the web page *without* CSS, which is the case for many alternative browsers, including screen readers that read a page's content aloud to visually impaired visitors. Without CSS, all the sidebar material (which often includes navigational elements, ads, or other information that's not relevant to the main topic of the page) appears before the content the visitor came to read in the first place. The inconvenience of having to scroll past the same sidebar content on each page will turn off some visitors. Furthermore, your page is less accessible to vision-impaired visitors, who have to listen to their screen readers read off a long list of links and ads before coming to any real information.

And if that doesn't sway you, you've got the search engines to worry about. Most search engines limit the amount of HTML they read when searching a site. On a particularly long web page, they simply stop at a certain point—possibly missing important content that *should* be indexed by the search engine. Also, most search engines give greater value to the HTML near the beginning of the file. So if you're worried about getting good placement in search engine results, it's in your best

interest to make sure the important content is as close as possible to the top of the page's HTML code.

In the top-left diagram in Figure 13-4, the main content's HTML is between the left and right sidebars, which is better than having it after both sidebars. You can even put the main content before *both* sidebars' HTML by wrapping the main content and left sidebar in one `<div>`, floating that `<div>` left, and then floating the main content right and the left sidebar left *within* that `<div>` (Figure 13-4, bottom). Voilà—the main column's HTML falls before the other `<div>` tags.

Floats Within Floats

The bottom diagram in Figure 13-4 illustrates another useful technique—floating elements *within* floats. Imagine that the main content (3) and the left sidebar (4) divs didn't exist, and only the column wrapper (2) and the right sidebar (5) were left. You'd have just a basic two-column design, with one column floated left and another floated right. In fact, it's still a two-column design even with the two divs (3 and 4) placed back inside the column-wrapper div. The difference is that the left column is itself divided into two columns.

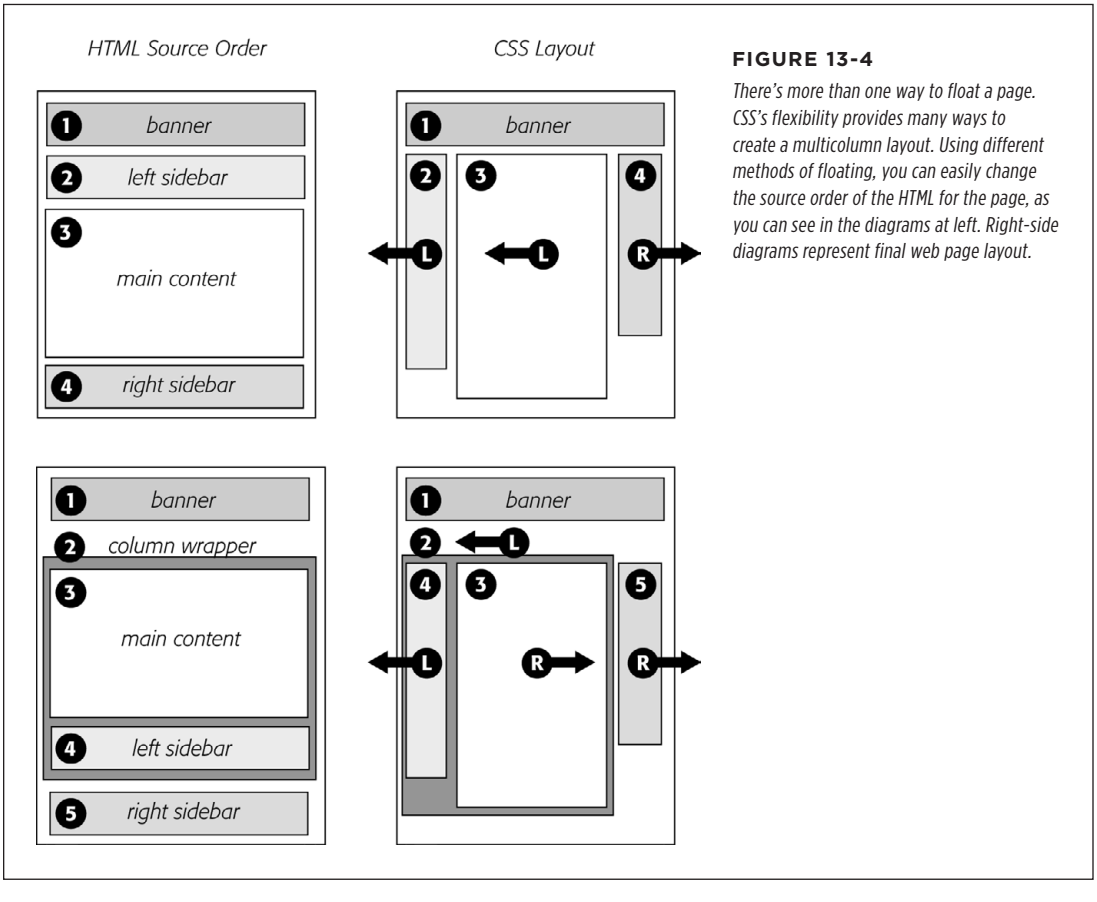

FIGURE 13-4

There's more than one way to float a page. CSS's flexibility provides many ways to create a multicolumn layout. Using different methods of floating, you can easily change the source order of the HTML for the page, as you can see in the diagrams at left. Right-side diagrams represent final web page layout.

Although this arrangement is a bit confusing, it's also helpful in a number of instances. First, it lets you add columns within a column. The three-column layout at the top of Figure 13-5 shows a small Tips box in the middle column that also has two columns inside it. By nesting floats inside floats, you can create some very complex designs.

In addition, when you have just a couple of floated elements divided into columns with additional floated elements, it's easier to calculate the widths of page elements. That's a good thing when you need to control float drops (page 480) and other problems that occur when columns get too wide.

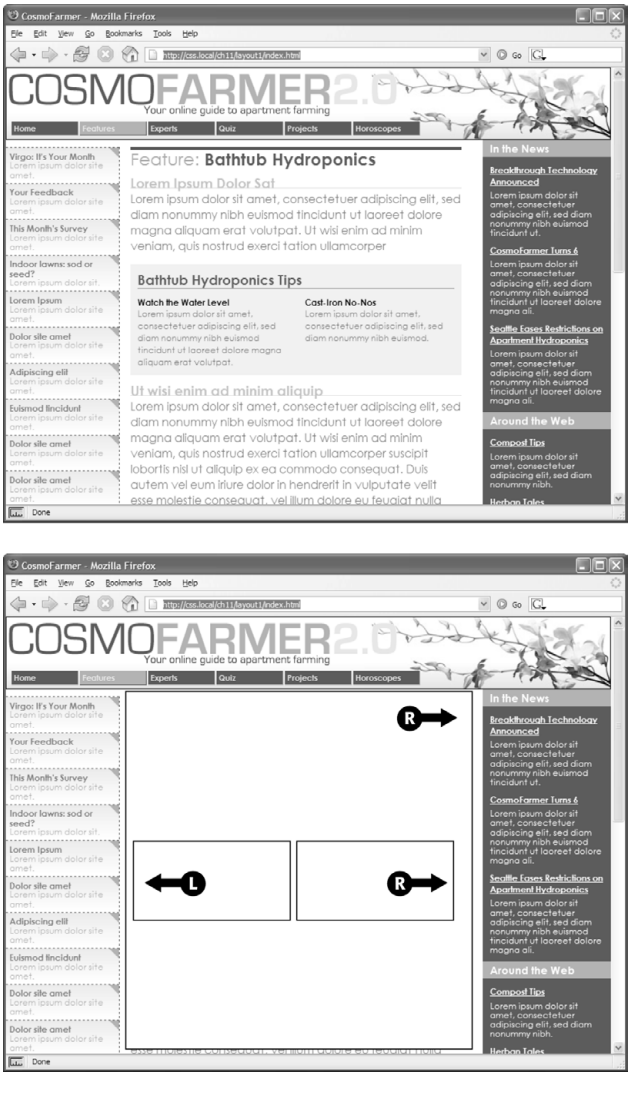

FIGURE 13-5

Top: Create columns within columns by floating elements inside another floated element. In the middle column, the Tips box provides a simple two-column note that adds visual interest to the page.

Bottom: It doesn't matter which direction the container is floated (in this instance, to the right)—you simply float the two additional columns left and right.