CHAPTER 6

Page Layout with CSS

Chapter Objectives

In this chapter, you will learn how to ...

- Describe reasons to use CSS for page layout
- Use relative and absolute positioning
- Apply the CSS Box Model
- Configure basic page layouts using CSS
- Configure two-column page layouts using CSS
- Locate CSS page layout resources

Now that you are familiar with using CSS to format text and color, you are ready to explore using CSS to configure Web page layout. This method relies on CSS properties rather than tables to design a Web page. The technology for this layout is called CSS-P, for CSS positioning. This chapter introduces you to configuring page layouts using CSS.

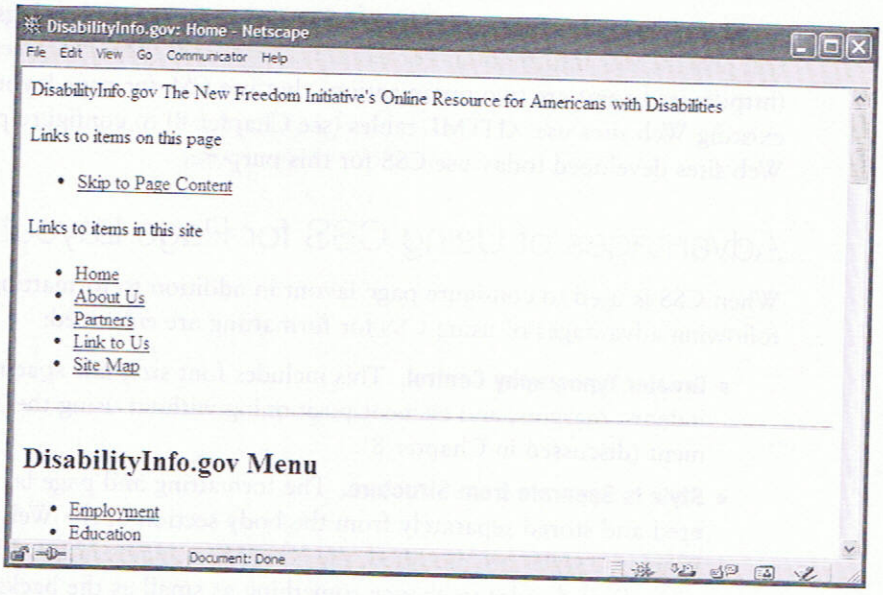


Figure 6.2 The DisabilityInfo.gov home page displayed in Netscape 4.79, an outdated browser

it is coded in the Web page. Because the site configures most of its graphics as background images within the external CSS style sheet, these graphics are unavailable to the older browser. The visitor experience is not exactly compelling when using an out-of-date browser. However, the site can still be navigated and information can be obtained.

Figure 6.3 shows the same page displayed in an Opera browser configured to simulate a text browser. The text content is displayed in the order it is coded in the Web page. Notice how even with the newer CSS page layout techniques utilized, the page is usable when rendered in a text browser.

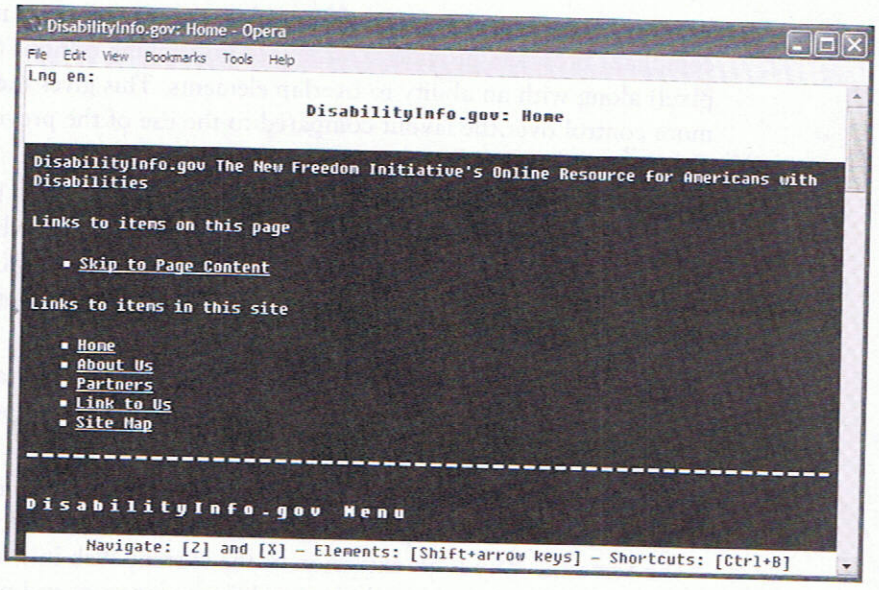


Figure 6.3 A text browser simulation of DisabilityInfo.gov's home page

Web developers have long delayed using CSS for page layout because of these rendering issues. What has recently changed? As modern browser support of CSS has increased, the number of Web visitors who use older browsers has decreased. For a typical Web site, less than 1 percent of visitors use these older browsers. Depending on your site's

Disadvantages of Using CSS for Page Layout

If you review the screenshots of a site that uses CSS for page layout (Figures 6.1 and 6.2) you'll see a very obvious disadvantage. Visitors using older browsers will not experience your Web site in the same way as visitors using modern browsers. If you are using tables for page layout, this is not an issue. Why then are developers beginning to code mainstream Web sites using CSS for page layout? With good support of CSS by modern browsers and the increasing use of modern browsers, the advantages of using CSS to configure page layout usually outweigh the disadvantages. Of course, the target audience of a Web site should be a deciding factor. For example, if your target audience for an intranet site is a company that has standardized on Netscape 4.7 for the desktop, none of the advantages would be realized and it would be better to design the site using tables for page layout. The projected (and eventually actual) target audience should be considered when deciding on a page layout technique.

Even with the increased CSS support of modern browsers, there are still differences and bugs in their implementation of the W3C Recommendations. This is a disadvantage for Web developers, since coding and testing time is increased. Leading developers have created Web sites that document and discuss these issues (see http://www.quirksmode.org and http://www.positioniseverything.net). The CSS techniques in this chapter have been tested with Internet Explorer 7, Opera 9.21, Firefox 2, and Apple's Safari for Windows beta.

Another potential disadvantage is the fact that experienced Web developers who are adept at coding pages using XHTML tables for layout will see productivity drop temporarily as they learn about CSS techniques and properties. Using CSS positioning is different from configuring pages with tables. Time and practice are needed when learning something new.

At this point you've seen some examples of using CSS for page layout and are aware of the issues related to using this technology. The next section discusses the **CSS Box Model**—a crucial building block of CSS positioning.

FAQ

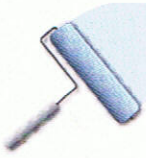
Should XHTML tables never be used?

Many commercial Web sites still use XHTML tables for page layout. This is for a very good reason—tables are widely supported by browsers. As a Web developer you will most likely work on sites that use XHTML layout tables and you'll work with these in Chapter 8. However, a growing trend is to configure pages using CSS (sometimes called table-less layout). This does not mean that tables are bad, ineffective, or that they are never coded on Web sites that use CSS for page layout. Even Web sites with so-called "table-less" layouts may include tables to present information in a tabular manner or facilitate design of a small portion of the page.

6.2 The Box Model

Each element in a document is considered to be a rectangular box. As shown in Figure 6.4, this box consists of a content area surrounded by padding, a border, and margins. This is known as the Box Model.

browser used **normal flow** (the default) and displayed the elements in the order that they appeared in the source code. As you've worked through the exercises in the previous chapters, you created Web pages that the browser rendered using normal flow. You'll practice this a bit more in the next Hands-On Practice, then later in the chapter you'll experiment with positioning to configure the flow, or placement, of elements on a Web page.



HANDS-ON PRACTICE 6.1

You will explore the box model in this Hands-On Practice as you create the Web pages shown in Figure 6.5 and Figure 6.6.

Practice with Normal Flow

Launch a text editor and open the Chapter6/starter1.html file from the student files. Save the file with the name box1.html. This page is displayed in Figure 6.5. Edit the body of the Web page and add the following code to configure two `<div>` elements:

```
<div class="div1">
This is the first box.
</div>
<div class="div2">
This is the second box.
</div>
```

Now let's add the CSS to configure the "boxes." Add a new style rule for a class named `div1` to configure a light gray background, dashed border, width of 200, height of 200, and 5 pixels of padding. The code follows:

```
.div1 {  width: 200px;
         height: 200px;
         background-color: #cccccc;
         border: dashed;
         padding: 5px;
}
```

Create a style rule for a class named `div2` to configure a width and height of 100, ridged border, 10 pixel margin, and 5 pixels of padding. The code follows:

```
.div2 {  width: 100px;
         height: 100px;
         background-color: #ffffff;
         border: ridge;
         padding: 5px;
         margin: 10px;
}
```

Save the file. Launch a browser and test your page. It should look similar to the one shown in Figure 6.5. The student files contain a sample solution at Chapter6/box1.html.

Figure 6.7 shows a Web page (see the student files, Chapter6/relative.html) that uses relative positioning and the `left` property to configure the placement of a `<div>` (assigned to the `id mycontent`) to the left of the normal flow.

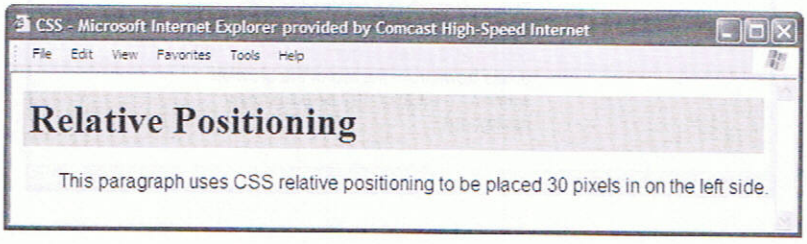


Figure 6.7
The paragraph is configured using relative positioning

The result is that the content of the `<div>`—the paragraph—is rendered 30 pixels in from the left where it would normally be placed at the browser's left margin. W3C Recommendations call for positioning to be applied to any element and Internet Explorer follows this recommendation. However, cross-browser support of positioning is more reliable when the `<div>` element is used for positioning. Notice also how the `padding` and `background-color` properties configure the heading element. The CSS follows:

```
#myContent { position: relative;
             left: 30px;
             font-family: Arial,sans-serif;
}
h1 { background-color: #cccccc;
     padding: 5px;
     color: #000000;
}
```

The XHTML source code follows:

```
<h1>Relative Positioning</h1>
<div id="myContent">
  <p>This paragraph uses CSS relative positioning to be placed 30
pixels in on the left side.</p>
</div>
```

Use **absolute positioning** to specify the location of an element precisely in a browser window. The `position:absolute` property along with either a `left`, `right`, and/or `top` property is needed to configure the placement.

Figure 6.8 shows a screenshot of a Web page that uses absolute positioning to configure a `<div>` (see the student files, Chapter6/absolute.html).

The `<div>` is assigned to the `content id` which is positioned 200 pixels in from the left margin and 100 pixels down from the top of the browser window. The result is that the paragraph contained within the `<div>` is rendered 200 pixels in from the left side and 100 pixels down from the top of the document area in the browser window. The width of the `<div>` is set to 300 pixels. Again, `padding` and `background-color` are used to configure the heading element. The CSS follows:

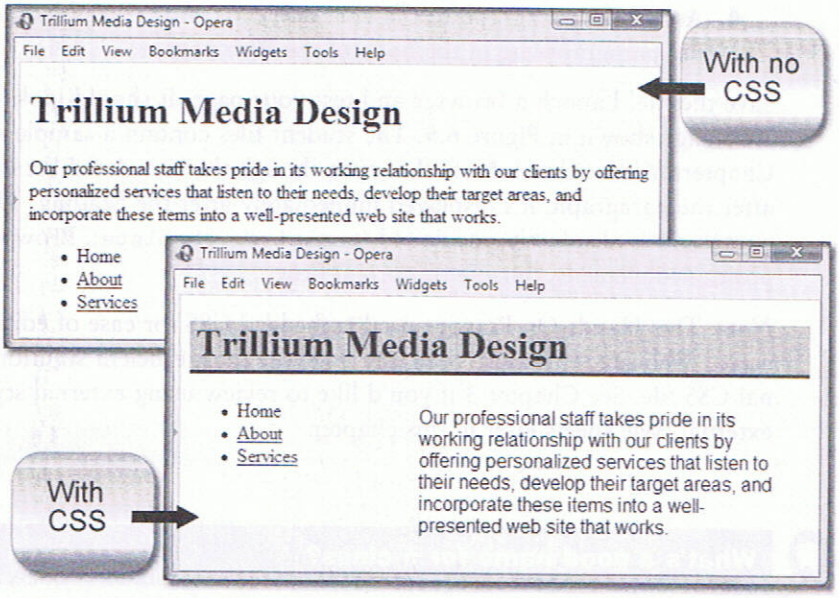


Figure 6.9
The lower Web page uses CSS absolute positioning

Let's add the CSS to make this page more "stylish" and look like the lower screenshot in Figure 6.9. Save the file with the name trillium.html. With trillium.html open in a text editor, modify the code as follows:

1. This page uses embedded styles. Code opening and closing `<style>` tags in the header section.

```
<style type="text/css">
</style>
```

2. Create style rules for the `h1` selector. Configure a background color (`#B0C4DE`), text color (`#000080`), a 3 pixel solid bottom border in the color `#000080`, and 5 pixels of padding on the bottom and left sides.

```
h1 { border-bottom: 3px solid #000080;
     color: #000080;
     background-color: #B0C4DE;
     padding: 0 0 5px 5px;
}
```

Note: The padding can be set for each side individually using the `padding-top`, `padding-right`, `padding-bottom`, and `padding-left` properties. You can use shorthand notation to set all four values in one `padding` property. The order of the numeric values determines which box side is configured (top, right, bottom, left).

3. Create style rules for a class named `content`. Configure the position to be absolute, 200 pixels from the left, 75 pixels from the top, a width of 300 pixels, and Arial or sans serif font typeface.

```
.content { position: absolute;
           left: 200px;
           top: 75px;
           font-family: Arial,sans-serif;
           width: 300px;
}
```

```
h1 { background-color: #cccccc;
     padding: 5px;
     color: #000000;
}
p { font-family:Arial,sans-serif;
}
#yls { float: right;
       margin: 0 0 5px 5px;
       border: solid;
}
```

The XHTML source code follows:

```
<img id="yls" src="yls.jpg" alt="Yellow Lady Slipper" height="100"
width="100" />
<h1>Floating an Image</h1>
<p>The heading and paragraph follow normal flow. The Yellow Lady
Slipper pictured on the right is a wildflower. It grows in wooded
areas and blooms in June each year. The Yellow Lady Slipper is a
member of the orchid family.</p>
```

There are times when you want to clear the effect of the float. In these cases you use the `clear` property.

The Web page displayed in Figure 6.11 uses the `clear` property set to `clear:right`. The images each clear the float that precedes them and float to the right of the browser window (see the student files, Chapter6/float2.html).

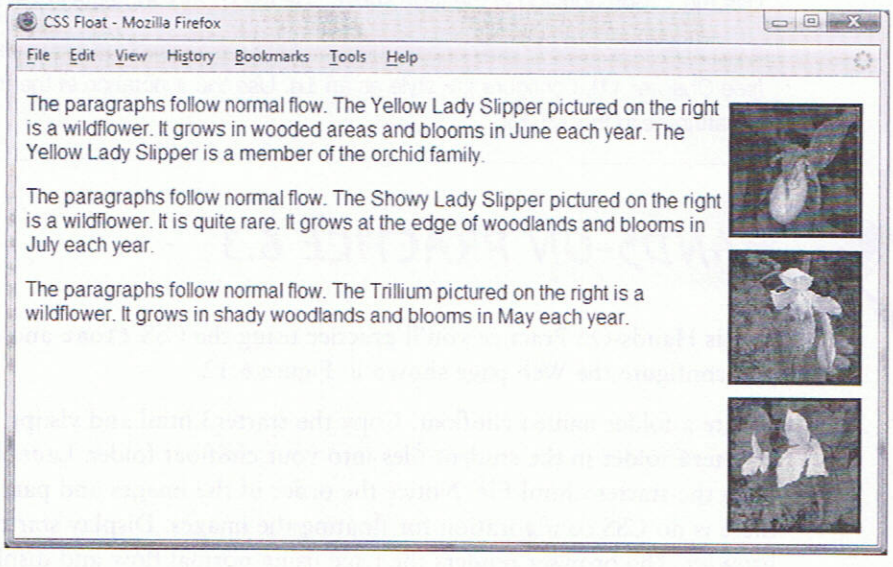


Figure 6.11
Using both the `float` and `clear` properties

The CSS follows:

```
.rightfloat { float: right;
              margin: 5px;
              clear: right;
              border: solid;
}
```

```
.float { float:left;
         margin-right:10px;
         border:ridge;
}
```

2. Assign the image element to the class named float (use `class="float"`).

Save the file. Launch a browser and test your page. It should look similar to the Web page shown in Figure 6.12. The student files contain a sample solution at Chapter6/floatyls.html.

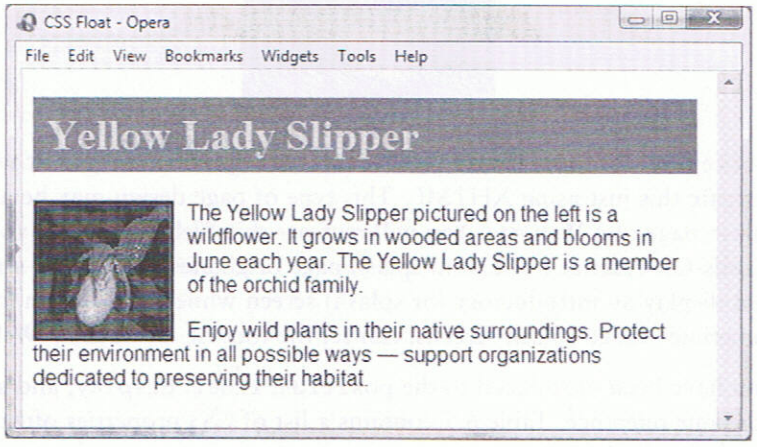


Figure 6.12
The CSS `float` property left-aligns the image

The `display` Property

Recall from Chapter 2 that some XHTML elements, such as the paragraph and heading elements, are block elements. A division (`<div>`) is also a block element. The browser renders these elements with 100 percent of the available width and displays a line break above and below—forming a "block." Other elements, such as anchor tags and span tags, are rendered directly inline—with no line break before or after them. These are called inline elements.

The **`display` property** configures if and how an element is displayed. An element configured with `display:none` will not be displayed. This is sometimes used when configuring styles to print a Web page. An element configured with `display:block` will be rendered as a block element (even if it is actually an inline element, such as an anchor tag). You will work with the `display` property in Chapter 7.

The `z-index` Property

The **`z-index` property** is used to modify the stacking order of elements on a Web page. When using only XHTML there is no easy way to "stack" elements other than configuring backgrounds for pages or tables. The `z-index` property provides flexibility in the display of elements. The default `z-index` value is `"0"`. Elements with higher `z-index` values will appear stacked on top of elements with lower `z-index` values rendered on the same position of the page. Figure 6.13 is configured using absolute positioning and `z-index` properties.

Table 6.1 CSS properties used with formatting and page layout (*continued*)

Property	Description	Commonly Used Values
`border-width`	Width of a border around an element	A numeric pixel value (such as `1px`), percentage value, or the text values `thin`, `medium`, `thick`
`clear`	Specifies the display of an element in relation to floating elements	Text values `left`, `right`, `both`, `none` (default)
`color`	Text color	Any valid color
`display`	Controls how and if the element will display	Text values `none`, `block`, `inline`, `list-item`. Display set to `"none"` causes the element not to display.
`font-family`	Name of a font or font family	Any valid font or a font family such as `serif`, `sans-serif`, `fantasy`, `monospace`, or `cursive`
`font-size`	Size of the text font	This varies; `pt` (standard font point sizes), `px` (pixels), the unit `em` (which corresponds to the width of the capital M of the current font), or percentages; the text values `xx-small`, `small`, `medium`, `large`, `x-large`, and `xx-large` are also valid
`font-style`	Style of the font	`normal` (default), `italic`, `oblique`
`font-weight`	Boldness or weight of the font	This varies: the text values `normal`, `bold`, `bolder`, and `lighter` can be used; the numeric values `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800`, and `900` can be used
`height`	Height of an element	A numeric value (`px` or `em`), numeric percentage, or `auto` (default)
`left`	Distance in from the left to display an element	A numeric pixel value or percentage
`line-height`	Spacing allowed for the line of text	It is most common to use a percentage for this value. For example, a value of `200%` is double space.
`list-style-image`	Image used to replace "bullets" in an XHTML list	`url(imagename.gif)` or `url(imagename.jpg)`
`list-style-type`	Indicates the type of list item marker	Text values `none`, `disc`, `circle`, `square`, `decimal`, `lower-roman`, `upper-roman`, `lower-alpha`, `upper-alpha`
`margin`	Shorthand notation to configure the margin surrounding an element	A numeric value (`px` or `em`) or percentage; for example: `body { margin: 10px;}` will set page margins in the document to 10 pixels. If you set a value to 0 pixels, omit the `px`. Four numeric values (`px` or `em`) can be specified. The values configure the margins in the following order (`margin-top`, `margin-right`, `margin-bottom`, `margin-left`).

continued

Table 6.1 CSS properties used with formatting and page layout (*continued*)

Property	Description	Commonly Used Values
`text-indent`	Indents first line of a block element	A numeric value (`em` or `px`), percentage
`text-transform`	Modifies appearance of text	Text values `none` (default), `capitalize`, `uppercase`, `lowercase`
`top`	Distance down from the top to display an element	A numeric pixel value or percentage
`vertical-align`	Modifies the alignment of an inline element	Text values `middle`, `bottom`, `text-bottom`, `text-top`, `top`, `super`, `sub`, or a percentage value
`visibility`	Controls whether an element displays and takes up space on a Web page	Text values `visible`, `hidden`, `inherit`
`width`	Width of an element	A numeric value (`px` or `em`), percentage
`z-index`	The stack order of an element on a Web page; a higher value will display in front of elements with lower values	A numeric value; the default value is 0. May be negative although this can cause problems in Netscape.

HANDS-ON PRACTICE 6.4

Of course, the best way to learn new coding technologies is to practice them. In this Hands-On Practice you will configure two files: an external style sheet (wildflower.css), and a splash page similar to the one shown in Figure 6.13. You will use CSS to format and position the page elements.

Getting Started

Locate the yls.jpg, pls.jpg, showy.jpg, and trillium.jpg files in the Chapter6 folder in the student files. Create a new folder called wildflowers. Copy the files to the wildflowers folder.

Part 1—Code the Splash Page

Review Figure 6.13 and notice the page elements: three images (yls.jpg, pls.jpg, and trillium.jpg), a logo, and a link. Figure 6.14 shows a wireframe of these elements arranged on the page.

In this part of the Hands-On Practice you will code each page element using XHTML and wrap it in a `<div>` that is assigned to an `id`. Then, in Part 2 you will code CSS to configure a number of properties including the absolute position, border, font-family, and so on. As you code the splash page, splash.html, you will place the elements on the page and assign `id` values. In essence, you are configuring a set of boxes (using the box model). Launch Notepad and type in the following XHTML:

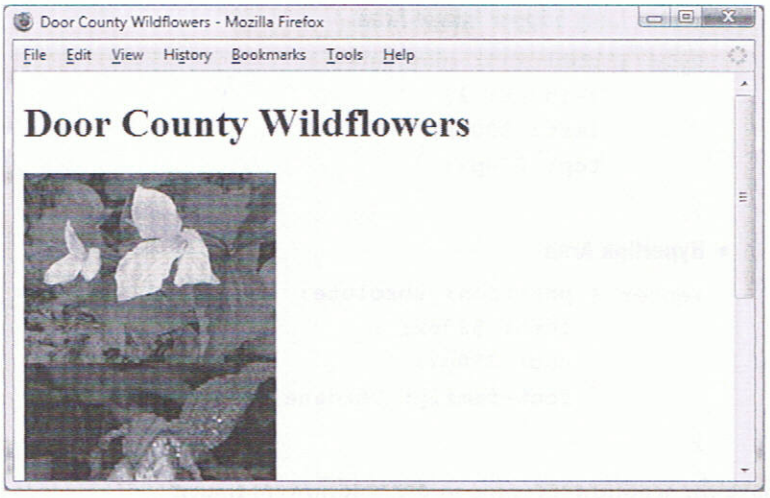


Figure 6.15
The splash page before the styles are coded

Figure 6.14 and to the screenshot shown in Figure 6.13. Type the CSS in your wildflower.css file as the styles are discussed as follows:

- **Splash Screen Logo Area.** This should be configured with the `id` value of `splashlogo`. This `id` will use absolute positioning, appear 210 pixels from the top of the browser window, use the background color of `#e8b9e8`, use a text color of black, display in Times New Roman or serif font with center-aligned text, have about 5 pixels of padding on the top and bottom sides, and have 20 pixels of padding on the left and right sides. There should be a 2 pixel solid border. This element will overlap the images so it must have a higher `z-index` value than the three images. The CSS follows:

```
#splashlogo { background-color: #e8b9e8;
              padding: 5px 20px;
              color: #000000;
              font-family: "Times New Roman", serif;
              position: absolute;
              text-align: center;
              z-index: 4;
              top: 210px;
              left: 80px;
              border: 2px solid #000000;
}
```

- **Trillium Image Area**

```
#trillium { position: absolute;
            z-index: 3;
            left: 20px;
            top: 80px;
}
```

- **Pink Lady Slipper Image Area**

```
#pls { position: absolute;
       z-index: 2;
       left: 420px;
       top: 130px;
}
```

with the `float` property appear to float on the right or left side of either the browser window or another XHTML element such as a paragraph. For best results, elements that float should have an intrinsic width (such as an image element) or have their width configured. In the next Hands-On Practice you'll gain more practice using floats.

HANDS-ON PRACTICE 6.5

In the previous Hands-On Practice you created the splash page for the Door County Wildflowers Web site. In this Hands-On Practice you'll add to this site—create a new content page (page1.html) and modify the wildflower.css external CSS file to configure page1.html to display similar to the one shown in Figure 6.16 (shown also in the color insert section).

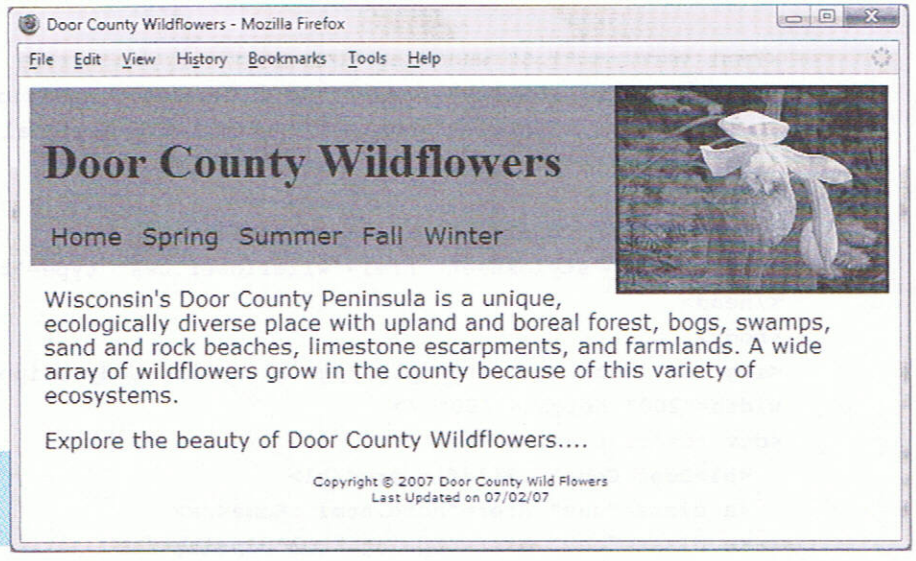


Figure 6.16
The image is floating on the page

See the center color insert

Getting Started

Locate the wildflowers folder you created in the previous Hands-On Practice. Copy the showy.jpg from the Chapter6 folder in the student files to the wildflowers folder.

Part 1—Code the Content Page

Review Figure 6.16 and notice the page elements: the logo, navigation area, floating right image, page content, and page footer. Figure 6.17 shows a wireframe sketch of these elements on the page.

In this part of the Hands-On Practice, you will code each element in an XHTML document. These will be coded to use `id`s and `class`es that correspond to CSS, which configures a number of properties including the `margin`, `padding`, `border`, `font-family`, and so on. As you code the content page, page1.html, you will place the elements on the page and assign `id` and `class` values. You are creating a combination of nested boxes, instead of coding tables and cells. Launch Notepad and type in the following XHTML:

Save your page in the wildflowers folder and test it in a browser. It will not look like Figure 6.16 since you have not yet configured all the ids and classes in the external style sheet. Your page should look similar to the one shown in Figure 6.18.

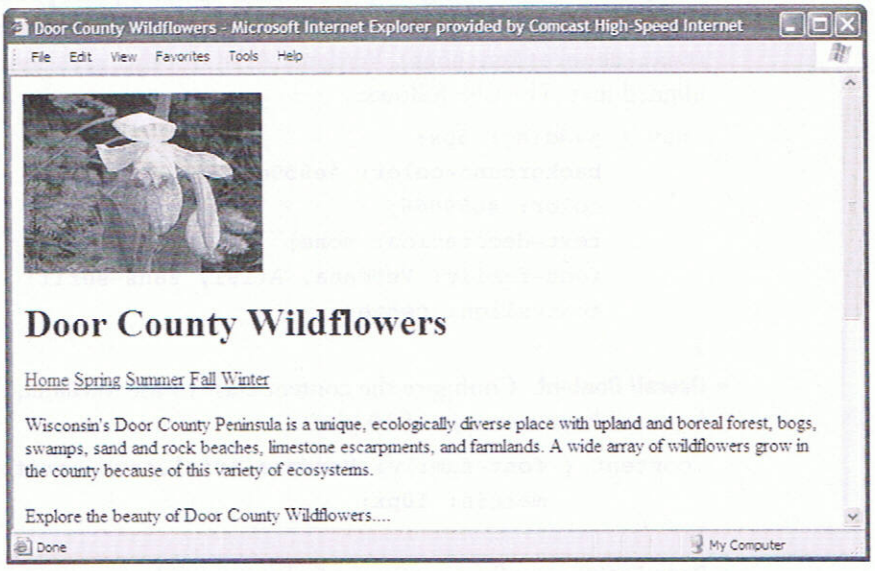


Figure 6.18
The page before CSS for positioning is configured

Part 2—Add Styles to the External Style Sheet

Open your wildflower.css file with Notepad and prepare to add additional styles to configure the page1.html page. Let's take a moment to consider what type of positioning is needed for the page shown in Figure 6.16, the page1.html page. Locate the following areas on the sketch in Figure 6.17: logo area, logo image, navigation links, content, and footer. Type the CSS in your wildflower.css file as the styles are discussed as follows:

- **Logo Area.** This should be configured with the `id` value of `contentlogo`. Configure this `id` so that the `background-color` is `#e8b9e8`, the text color is black, the `font-size` is larger, and the padding is 10 pixels. The CSS to configure this follows:

```
#contentlogo { background-color: #e8b9e8;
               font-size: larger;
               padding: 10px;
               color: #000000;
}
```

- **Logo Image.** Notice how this image floats at the right edge of the browser window. Configure the id `imagelogo` with a right float. We have most often set the margin to be the same for all sides of an element's box. The margins can be set for each side individually using the `margin-top`, `margin-right`, `margin-bottom`, and `margin-left` properties. A short-hand version of this is to set all four values in one `margin` property. The order of the numeric values determines which box side is configured (top, right, bottom, left). In this page layout the margin at the top and right of the `imagelogo` should be set to 0; the bottom and left margins should be set to 5 pixels. Configure a solid border. The CSS follows:

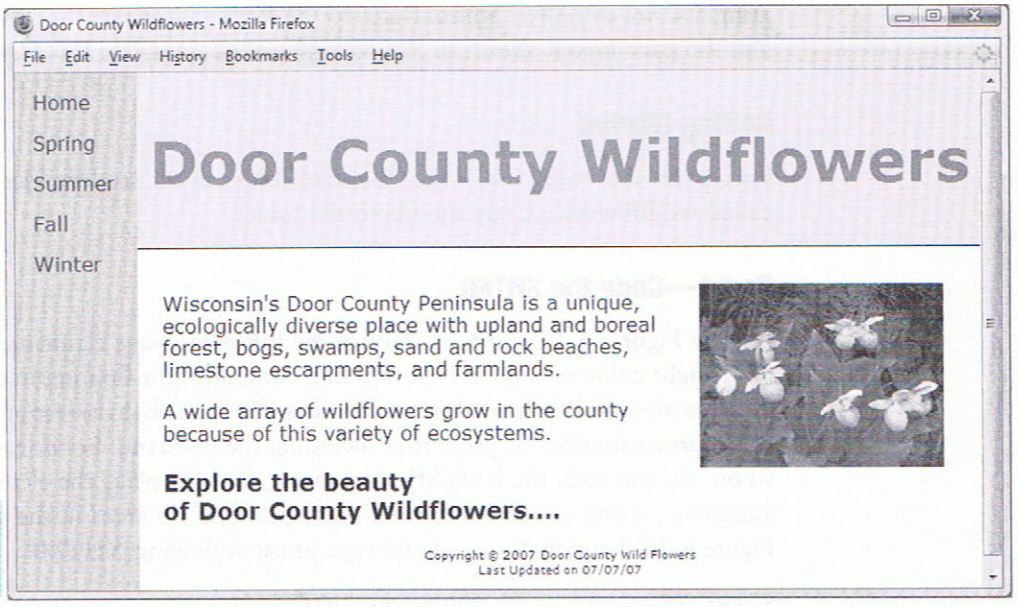


Figure 6.19
A two-column page configured using CSS

See the center color insert

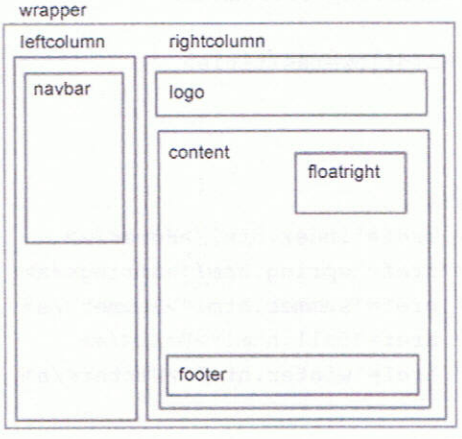


Figure 6.20
The two-column page layout

The page is designed with a number of boxes that correspond to the following page areas: wrapper, left column, right column, logo, navigation bar, content, right-floating image, and footer. Each of these areas will correspond to a `class` or `id` configured using CSS. The key to this layout is that the left column is coded to float to the left using `float:left`. With the left column navigation area floating to the left, the browser renders the other content down the page using normal flow. In the next Hands-On Practice you will code the XHTML and CSS to create the page shown in Figure 6.19.

HANDS-ON PRACTICE 6.6

In this Hands-On Practice you will develop your first two-column Web page using CSS. As you create the Web page and CSS, keep in mind that using this technique is like cre-

```
        Copyright &copy; 2007 Door County Wild Flowers<br />
        Last Updated on 07/07/07
      </div>
    </div>
</div>
</body>
</html>
```

Save your page as twocolumn.html in your wildflowers2 folder. Test the page in a browser. Your display will not look like the one shown in Figure 6.19 because you have not yet configured the CSS. Your page should look similar to the page shown in Figure 6.21.

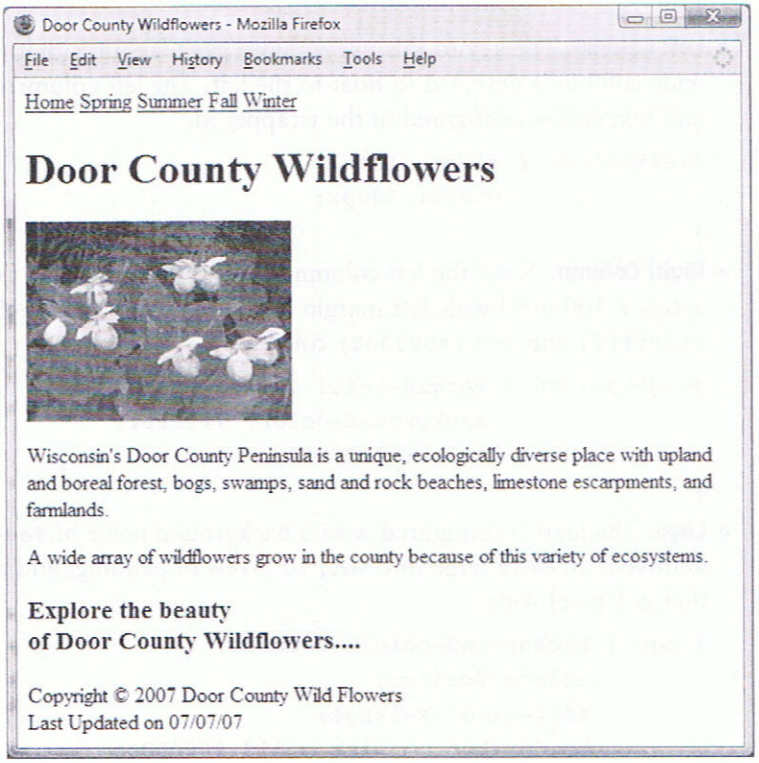


Figure 6.21
The two-column page before CSS is applied

Part 2—Configure the CSS

For ease of editing, in this Hands-On Practice you will code the CSS as embedded styles in the header section of the Web page. However, if you were creating an entire Web site you would most likely use an external style sheet as you did in the previous Hands-On Practice.

Launch Notepad and open twocolumn.html. Let's take a moment to consider what type of layout is needed for the page shown in Figure 6.19: wrapper, left column, right column, logo, navigation bar, content, right-floating image, and footer. Locate these areas on the sketch shown in Figure 6.20. Notice that the same font is used throughout the page and the page begins right at the browser margin. Launch Notepad and open your twocolumn.html file. In the header section of your Web page document, add a tag to begin the embedded styles: `<style type="text/css">`.

- **Image Floating at the Right.** Configure the image with a 10 pixel margin and float:right.

```
#floatright { margin: 10px;
              float: right;
}
```

- **Footer.** Configure the page footer with very small text that is centered. A `clear:right` is needed to clear the float of the image. Configure 20 pixels of padding on the bottom of the footer.

```
.footer { font-size: xx-small;
          text-align: center;
          clear: right;
          padding-bottom: 20px;
}
```

- **Navigation Area.** Configure the class for the navigation with no underlines (`text-decoration:none`), a 15 pixel margin, and to use `display:block`, which will allow each anchor tag act as a block element and be displayed on a separate line.

```
.navBar { text-decoration: none;
          margin: 15px;
          display: block;
}
```

Type the closing XHTML style tag as follows.

```
</style>
```

Save the twocolumn.html file in the wildflowers2 folder.

Part 3—Test the Page

Now that your styles are coded, test the twocolumn.html page again. Your page should be similar to the screenshot shown in Figure 6.19. If there are differences, verify the `id` and `class` values in your XHTML and check the syntax of your CSS. You may find the W3C CSS validator at http://jigsaw.w3.org/css-validator helpful when verifying CSS syntax. The student files contain a copy of twocolumn.html in the Chapter6/wildflowers2 folder.

This is just one of many ways that a two-column page layout can be coded. The best way to learn is to experiment by changing some of the properties and noting the result. It is a very good idea to test your pages in more than one browser. The pages in this chapter were tested using Internet Explorer 7, Opera 9, Safari for Windows beta, and Firefox 2.

- **Configure Temporary Background Colors.** Sometimes your code is valid but the browser window is not configured the way you would expect. If you temporarily assign distinctive background colors such as red or yellow and test again, it should be easier to see where the "boxes" are ending up.
- **Configure Temporary Borders.** Similar to the temporary background colors, you could temporarily configure an element with a 3 pixel red solid border—this will really jump out at you and help you recognize the issue sooner.
- **Use Comments to Find the Unexpected Cascade.** Styles and XHTML attributes configured farther down the page can override earlier styles. If your styles are misbehaving, try commenting out (see below) some styles and test with a smaller group of statements. Then add the styles back in one by one to see where or when the breakdown occurs. Work patiently and test the entire style sheet in this manner.

Note that Comment Areas Are Ignored by Browsers. A style sheet comment begins with `/*` and ends with `*/`. Comments can span multiple lines. A code snippet with CSS comments follows:

```
/* Set Page Margins to Zero */
body { margin: 0
}
/* temporarily commented out during testing
.nav { text-decoration: none;
}
*/
```

The first comment is used to document the style sheet and describe the style applied to the body tag. The second comment spans multiple lines. It begins on the line above the `nav` class and ends on the line below the `nav` class. This causes the browser to skip the `nav` class when applying the style sheet. This technique can be useful in testing when you are experimenting with a number of properties.

6.7 CSS Page Layout Resources

This chapter introduces you to using CSS for page layout configuration and should get you started in your exploration of this technology. It may help you to know that you are not alone in your quest to learn CSS. There are many resources with documentation, tutorials, and support for this technology. The page layout techniques discussed in this textbook are just an introduction to using this technology. There are many Web sites that offer additional insight and techniques for configuring page layout with CSS. The following are a few that you may find useful:

- http://glish.com/css

  Large collection of CSS page layouts and links to tutorials
- http://www.websitetips.com/css/index.shtml

  Comprehensive list of tutorials and CSS-related sites
- http://www.meyerweb.com/eric/css

  The site of Eric Meyer, a leading-edge Web developer

CHAPTER SUMMARY

This chapter introduced Cascading Style Sheet rules associated with page layout. Techniques for positioning and floating elements and configuring two-column page layouts were demonstrated. This topic is very deep and you have much to explore. Visit the resources in the chapter to continue learning about this technology.

Visit the textbook Web site at http://www.webdevfoundations.net for examples, the links listed in this chapter, and updated information.

Key Terms

absolute positioning
border
`clear` property
content
CSS Box Model
CSS-P
`display` property
`float` property
`left` property
margin
normal flow
padding
relative positioning
`right` property
`top` property
visible width
`z-index` property

Review Questions

Multiple Choice

1. When using absolute positioning, which of the following properties may be used to determine the placement of the element?
   a. `top` and `left`
   b. `z-index` and `display`
   c. `float` and `clear`
   d. none of the above

2. Which of the following, from outermost to innermost, are components of the box model?
   a. margin, border, padding, content
   b. content, padding, border, margin
   c. content, margin, padding, border
   d. margin, padding, border, content

3. Which of the following is the default value of the `border` and `padding` properties for an element?
   a. 1 pixel
   b. 0 pixels
   c. 3 pixels
   d. 10 pixels

4. Which of the following configures a class called `sidebar` to float to the right?
   a. `.sidebar { right: float; }`
   b. `.sidebar { float: right; }`
   c. `.sidebar { float-right: 200px; }`
   d. none of the above

5. Which of the following is the rendering flow used by a browser by default?
   a. XHTML flow
   b. normal display
   c. browser flow
   d. normal flow

6. Which of the following is used to change the location of an element slightly in relation to where it would otherwise appear on the page?
   a. the `float` property
   b. absolute positioning
   c. relative positioning
   d. this cannot be done with CSS

```
<body>
  <h1>CircleSoft Web Design</h1>
  <div id="content">
    <p>Our professional staff takes pride in its working
relationship with our clients by offering personalized services
that listen to their needs, develop their target areas, and
incorporate these items into a well-presented Web site that
works.</p>
  </div>
  <ul>
    <li class="nav">Home</li>
    <li class="nav"><a href="about.html">About</a></li>
    <li class="nav"><a href="services.html">Services</a></li>
  </ul>
  </div>
</body>
</html>
```

2. Fill in the Missing Code. This Web page should be configured as a two-column page layout with a right column 150 pixels wide. The right column should have a 1 pixel border. The padding in the left column content area needs to allow for the room that will be used by the right column. Some CSS properties and values, indicated by "_", are missing. Fill in the missing code.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Trillium Media Design</title>
<style type="text/css">
body { margin: 0;
       font-family: Verdana, Arial, sans-serif;
}
#rightcolumn { "_":"_";
               width: "_";
               background-color: #cccccc;
               height: 400px;
               border: "_";
}
#leftcolumn {
}
#logo { background-color: #cccccc;
        color: #663333;
        font-size: x-large;
        border-bottom: 1px solid #333333;
}
.content { padding: "_";
}
```

```
h1 { background-color: #eeeeee;
     padding: 5px;
     color: #666633;
     position: absolute;
     left: 200px;
     top: 20px;
}
p { font-family: Arial,sans-serif;
    position;
    absolute;
    left: 100px;
    top: 100px;
}
#yls { float:right;
       margin: 0 0 5px 5px;
       border: solid;
}
</style>
</head>
<body>
<img id="yls" src="yls.jpg" alt="Yellow Lady Slipper" height="100"
  width="100" />
<h1>Floating an Image</h1>
<p> The Yellow Lady Slipper pictured on the right is a wildflower.
It grows in wooded areas and blooms in June each year. The Yellow
Lady Slipper is a member of the orchid family.</p>
</body>
</html>
```

Hands-On Exercises

1. Write the CSS for a class with the following attributes: a light blue background color, Arial or sans-serif font, dark blue text color, 10 pixels of padding, and a narrow dashed border in a dark blue color.
2. Write the CSS for an id with the following attributes: float to the left of the page, light beige background, Verdana or sans-serif large font, and 20 pixels of padding.
3. Write the CSS to configure a class that will produce a headline with a dotted line underneath it. Choose a color that you like for the text and dotted line.
4. Write the CSS for an id that will be absolutely positioned on a page 20 pixels from the top and 40 pixels from the right. This area should have a light gray background and a solid border.
5. Write the CSS for a class that is relatively positioned. This class should appear 15 pixels in from the left. Configure the class to have a light green background.
6. **Extending Hands-On Practice 6.6.** Design a two-column page layout with the navigation on the right side. Use the twocolumn.html file from Hands-On Practice 6.6 as a starting point. This file is in the Chapter6/wildflowers2folder in the student files.

photos of the movie. Next, locate images from the movie. When you code your page use embedded CSS unless your instructor directs you otherwise. Hand in printouts of the moviesplash.html source code (print in Notepad), and the browser display of moviesplash.html to your instructor.


Focus on Ethics

(*Note*: It is unethical to steal an image from another Web site. Some Web sites have a link to their copyright policy. Most Web sites will give permission for you to use an image in a school assignment. If there is no available policy, e-mail the site's contact person and request permission to use the photo. If you are unable to obtain permission, you may substitute clip art or an image from a free site.)

Web Research

This chapter introduced using CSS to configure Web page layout. Use the resources listed in the text as a starting point. You can also use a search engine to search for CSS resources.

Create a Web page that provides a list of at least five CSS resources on the Web. For each CSS resource provide the URL, Web site name, and a brief description. Your Web page should use absolute positioning. Print both the source code (from Notepad) and the browser view of your Web page.

Focus on Web Design

There is still much for you to learn about CSS. A great place to learn about Web technology is on the Web itself. Use a search engine to search for CSS page layout tutorials. Choose a tutorial that is easy to read. Select a section that discusses a CSS technique that was not covered in this chapter. Create a Web page that uses this new technique. Consider how the suggested page layout follows (or does not follow) principles of design such as contrast, repetition, alignment, and proximity (see Chapter 5). The Web page should provide the URL of your tutorial, the name of the Web site, a description of the new technique you discovered, and a discussion of how the technique follows (or does not follow) principles of design. Print the external style sheet (if you used one), the Web page source code (from Notepad), and the browser view of your Web page.

WEB SITE CASE STUDY

Implementing CSS Two-Column Page Layout

Each of the following case studies continues throughout most of the text. This chapter implements CSS two-column page layout in the Web sites.

JavaJam Coffee House

See Chapter 2 for an introduction to the JavaJam Coffee House case. Figure 2.26 shows a site map for the JavaJam Web site. The pages were created in earlier chapters. In this

Figure 6.23
The new JavaJam two-column index.html

See the center color insert

pixel margin, and is displayed as a block element (with line breaks above and below) by the browser.

```
.navBar{ text-decoration: none;
         margin: 15px;
         display: block;
}
```

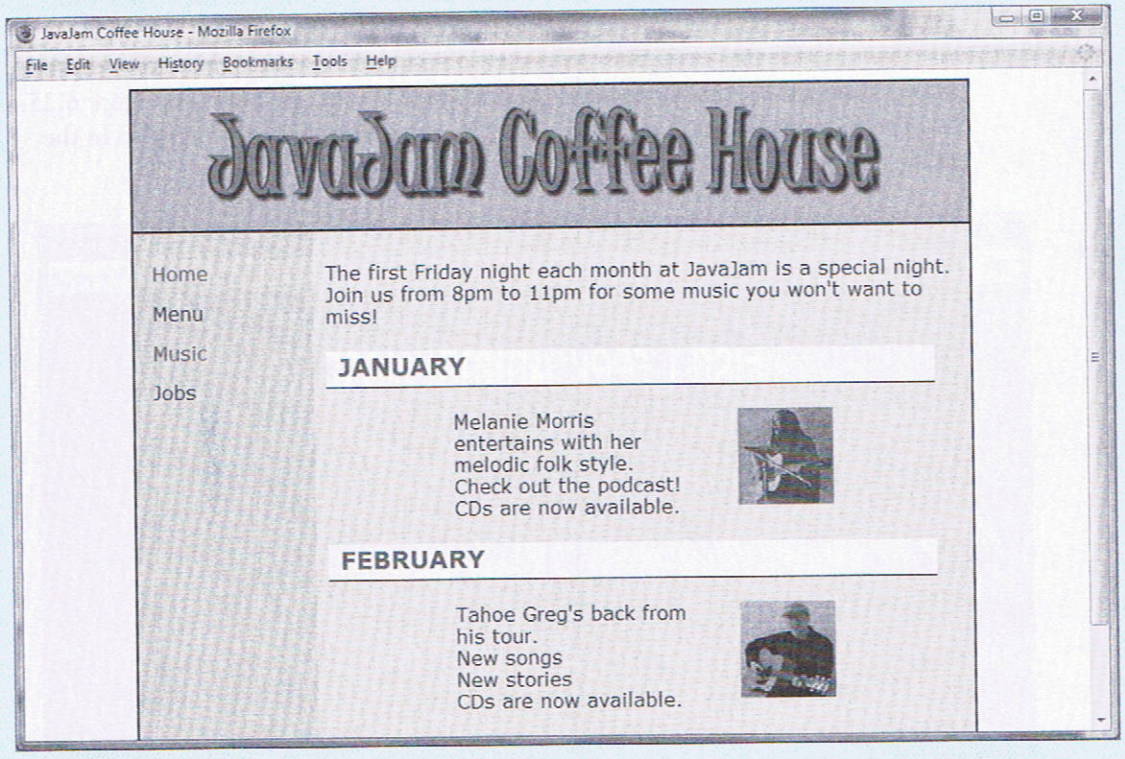
- Configure the right column area. Add a new style rule for the `rightcolumn` id to configure an area with a 150 pixel left margin, background (`#f1e8b0`) and text (`#000000`) color, and 10 pixels of padding.

```
#rightcolumn { margin-left: 150px;
               background-color: #f1e8b0;
               color: #000000;
               padding: 10px;
}
```

- Configure an area that floats to the right. Notice how the winding road graphic shown in Figure 6.23 floats on the right side—this is configured with the `floatright` class. Images are more compelling when separated from other elements (such as text) by empty space. Add 40 pixels of padding to the left side of this area.

```
.floatright { padding-left: 40px;
              float: right;
}
```

Figure 6.24
New style rules for the `h3` selectors

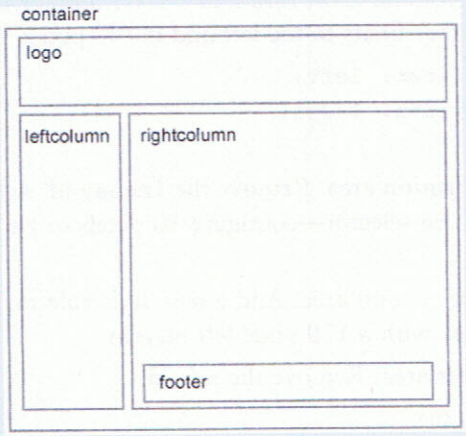


Fish Creek Animal Hospital

See Chapter 2 for an introduction to the Fish Creek Animal Hospital Case Study. Figure 2.30 shows a site map for the Fish Creek Web site. The pages were created in earlier chapters. In this case study you will implement a new two-column CSS page layout. You will modify the external style sheet and the Home, Services, and Ask the Vet pages. Unless your instructor directs you otherwise, use the Chapter 4 Fish Creek Web site as a starting point for this case study.

Figure 6.25 displays a wireframe for the two-column page layout with a page container, logo, left column, navigation, right column, and footer areas.

Figure 6.25
Fish Creek two-column page layout



3. **Modify the index.html File.** Add `<div>` elements and modify the code as follows:
   - Configure the logo area. Remove the `<h1>` opening and closing tags surrounding the Fish Creek image. Code a `<div>` that surrounds the Fish Creek logo image element. Assign the `<div>` to the id `logo`.
   - Configure the left column. The navigation image links are the only content in the left column. Change `<div id="imgnav">` to `<div id="leftcolumn">`.
   - Configure the right column. This area contains the definition list and the paragraph with the contact information. Code a `<div>` that surrounds this area. Assign the `<div>` to the id `rightcolumn`.
   - Configure the page footer area. You need to adjust the starting location of the `footer` id. Locate `<div id="footer">` in the code and remove the assignment to the id from the . Next, change `<div id="nav">` to `<div id="footer">`. The area assigned to the `footer` id now includes the text navigation, copyright information, and e-mail link.

   Save the index.html file. It should look similar to the Web page shown in Figure 6.26. Remember that validating your XHTML and CSS can help you find syntax errors. Test and correct this page before you continue.
4. **Modify the services.html and askvet.html Files.** Modify these Web page files in a similar manner as you did in Step 3. Save and test your pages in a browser. As you test your pages, use the CSS and XHTML validators to help you find syntax errors.

In this case study you changed the page layout of the Fish Creek Web site pages. Notice that with just a few changes in the CSS and XHTML code, you configured a two-column page layout.

Pete the Painter

See Chapter 2 for an introduction to the Pete the Painter Case Study. Figure 2.34 shows a site map for the Pete the Painter Web site. The pages were created in earlier chapters. In this case study you will implement a new two column CSS page layout for Pete the Painter. You will modify the external style sheet and of the Home, Services, and Testimonials, pages. Unless your instructor directs you otherwise, use the Chapter 4 Pete the Painter Web site as a starting point for this case study.

Figure 6.27 displays a wireframe for the two-column page layout with a page container, logo, left column, navigation, right column, and footer areas.

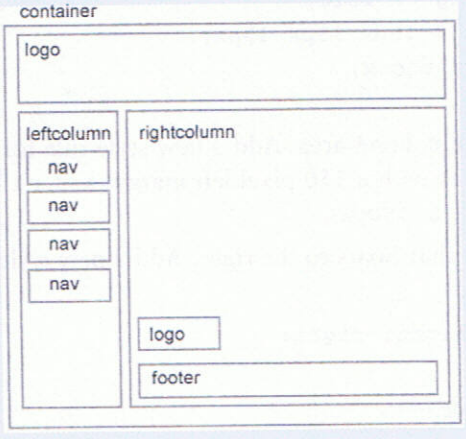


Figure 6.27
Pete the Painter two-column page layout

3. **Modify the index.html File.** Add `<div>` elements and modify the code as follows:
   - Configure the logo area. Remove the `<h1>` opening and closing tags surrounding the Pete the Painter image. Code a `<div>` that surrounds the Painter logo image element. Assign the `<div>` to the id `logo`.
   - Configure the left column. The navigation links are the only content in the left column. Change `<div id="nav">` to `<div id="leftcolumn">`. Assign each anchor element in this area to the `nav` class (`class="nav"`).
   - Configure the right column. This area contains the content (paragraph, unordered list, and heading 3 elements) and the footer section. Code a `<div>` that surrounds this area. Assign the `<div>` to the `rightcolumn` id.
   - Assign the `<h3>` element to the `logo` class.

   Save the index.html file. It should look similar to the Web page shown in Figure 6.28. Remember that validating your XHTML and CSS can help you find syntax errors. Test and correct this page before you continue.

4. **Modify the services.html and testimonials.html Files.** Modify these Web page files in a similar manner as you did in Step 3. Configure the room images on the testimonials.html page—on the opening image tag for each room photo, remove the `align="right"` attribute and add `class="floatright"`.

5. **Save and Test Your Pages in a Browser.** As you test your pages, use the CSS and XHTML validators to help you find syntax errors.

6. **Bonus Style.** Figure 6.29 shows an alternate design for the testimonials.html page. Notice how the `<h4>` elements are styled differently—it is set to float and is configured with a *negative* top margin. This allows the dark green box to stand out better on the page. The paragraph elements in this area are each assigned to a class that configures a top border of the same color green and extra padding.

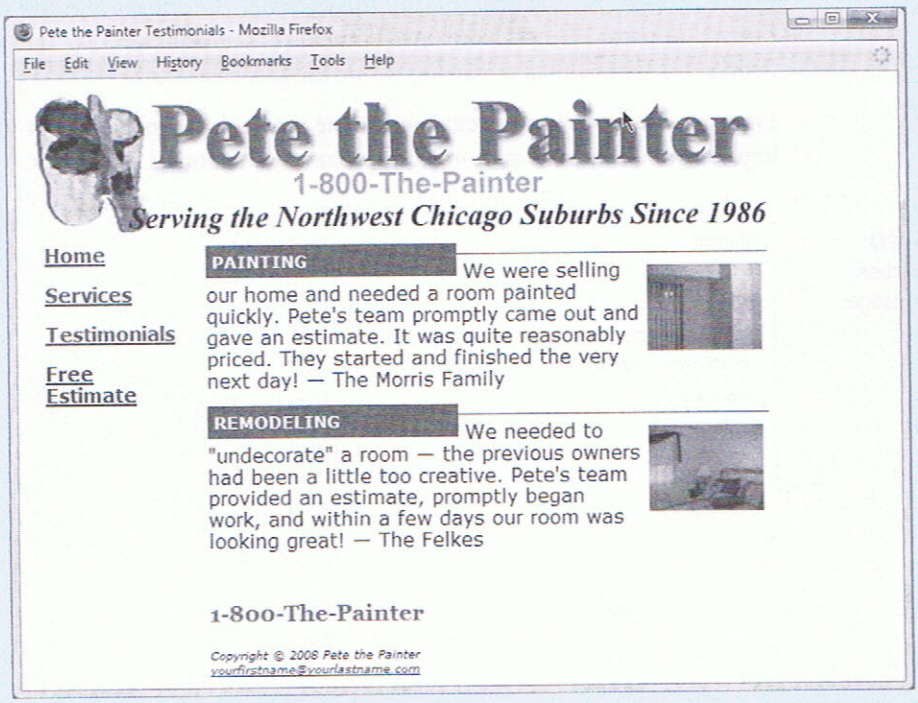


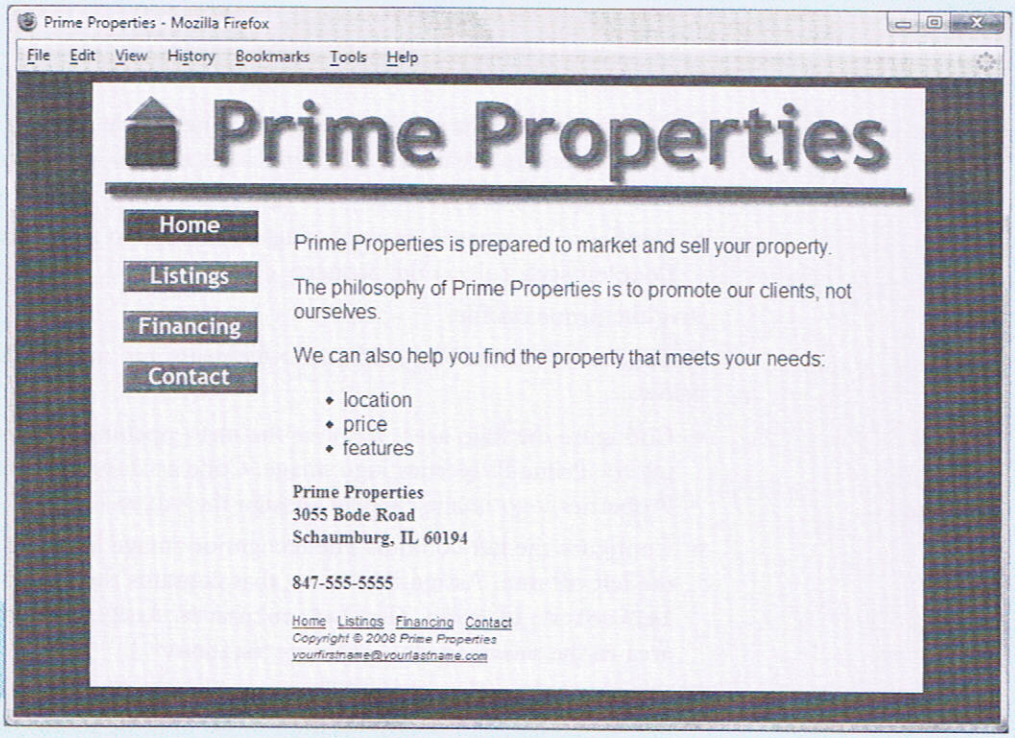
Figure 6.29
New style rules for the `h4` selector

Hands-On Practice Case

1. **Create a Folder.** Create a folder called primecss. Copy all the files from your Chapter 4 prime folder into the primecss folder. You will modify the prime.css file and each Web page file (index.html, listings.html, and financing.html) to implement the two-column page layout shown in Figure 6.30. See the new Prime Properties Home page, as shown in Figure 6.31 (shown also in the color insert section).

Figure 6.31
The new Prime Properties two-column index.html

See the center color insert



2. **Configure the CSS.** Open prime.css in Notepad. Edit the style rules as follows:
   - Configure the page background color. Modify the style rules for the `body` selector. Set the `background-color` property to `#003366`. Set the text color to `#ffffcc`.
   - Create a new `wrapper` id to contain the page content. Configure the area with a width of 680 pixels and centered (`margin:0 auto`). Set the minimum width to 680 pixels, also. Configure the background (`#ffffcc`) and text (`#003300`) colors. Set the left padding to 10 pixels.

```
#wrapper { width: 680px;
           min-width: 680px;
           margin: 0 auto;
           background-color: #ffffcc;
           color: #003300
           padding-left: 10px;
}
```

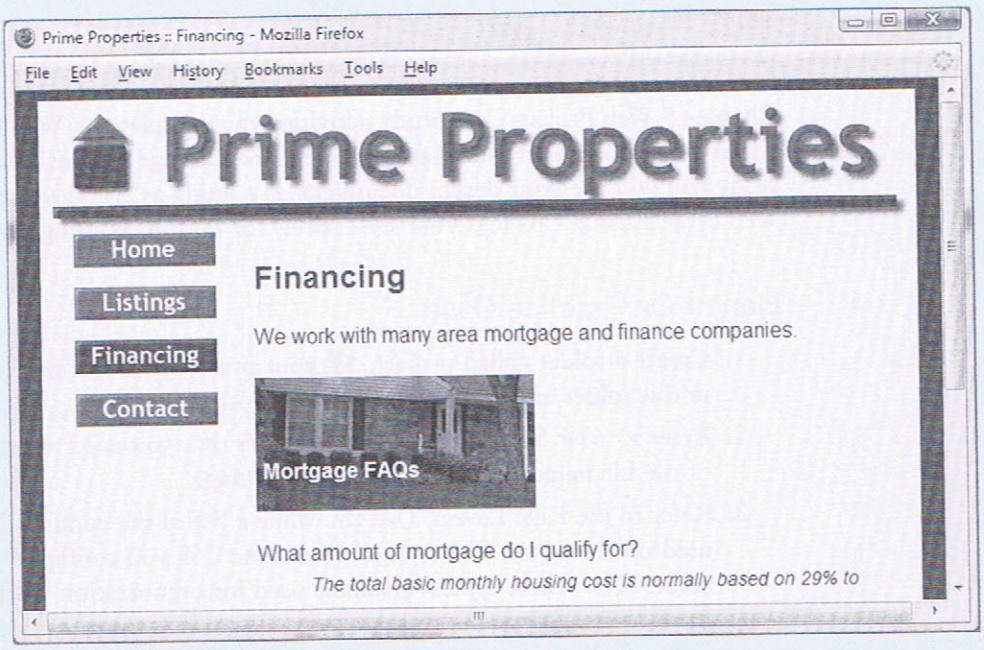


Figure 6.32
New style rules for the Mortgage FAQs heading

CSS creates the effect. Styles are declared for the `home` class with the following properties: a background image that does not repeat and is positioned very carefully, a width of 200 pixels, light text, and generous padding.

```
.home { background-image: url(schaumburg.jpg);
        background-position: -100px -260px;
        background-repeat: no-repeat;
        color: #ffffcc;
        padding: 60px 5px 20px 5px;
        width: 200px;
}
```

Notice that the `background-position` property is used with carefully chosen values. As indicated in Table 6.1, the `background-position` property can use two numeric pixel values—horizontal and vertical. The *negative numbers* cause the background image to shift 100 pixels to the left and 260 pixels down from the top. The effect is somewhat abstract and ends up displaying just part of the image. In this case, it is a part of a house to tie in with the home financing theme. Padding is set quite high to allow room for the image to display around the text. The text color is light in order to contrast well with the image.

Save the prime.css file. Modify the financing.html page as indicated above—add `class="home"` to the `<h4>` element. Save the financing.html file. Test your pages in a browser. Your financing.html page should look similar to the one shown in Figure 6.32. The other pages should display as they did at the end of Step 4.

In this case study you changed the page layout of the Prime Properties Web site pages. Notice that with just a few changes in the CSS and XHTML code, you configured a two-column page layout.