Open painter.css in a text editor and update the style rules with the following:

```
h4 { margin: -10px 5px 5px 0px;
     padding: 5px;
     font-family: Verdana, sans-serif;
     font-size: 14px;
     float: left;
     width: 200px;
     background-color: #336600;
     color: #ffffff;
     text-transform: uppercase;
}
.desc { padding: 5px 0 20px 0;
        border-top: 1px solid #336600;
}
```

Save the painter.css file. Test your pages in a browser. Your testimonials.html page should be similar to the one shown in Figure 6.29. The other pages should display as they did at the end of Step 4. Consider modifying the unordered list on the Services page (services.html) to use `<h4>` and `<p>` elements (assigned to the `desc` class) instead—the result will be a more cohesive design for your Web site.

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

Prime Properties

See Chapter 2 for an introduction to the Prime Properties Case Study. Figure 2.38 shows a site map for the Prime Properties Web site. The pages were created in earlier chapters. In this case study you will implement a new two-column CSS page layout for Prime Properties. You will modify the external style sheet and the Home, Listings, and Financing pages. Unless your instructor directs you otherwise, use the Chapter 4 Prime Properties Web site as a starting point for this case study.

Figure 6.30 displays a wireframe for the two-column page layout with a page wrapper, logo, left column, navigation, right column, and footer areas.

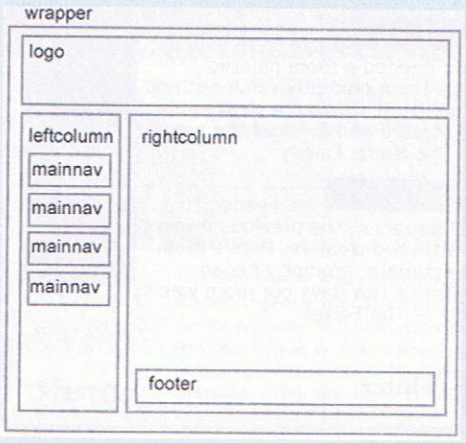


Figure 6.30 Prime Properties two-column page layout

target audience, this figure could be higher or lower—your Web logs (see Chapter 13) will provide this information. For example, TruGreen (http://trugreen.com) and wired (http://wired.com) are two organizations that use CSS for page layout. Although many existing Web sites use XHTML tables (see Chapter 8) to configure page layout, most Web sites developed today use CSS for this purpose.

Advantages of Using CSS for Page Layout

When CSS is used to configure page layout in addition to formatting text and color, the following advantages of using CSS for formatting are enhanced:

- **Greater Typography Control.** This includes font size, line spacing, letter spacing, indents, margins, and element positioning without using the XHTML table element (discussed in Chapter 8).
- **Style Is Separate from Structure.** The formatting and page layout can be configured and stored separately from the body section of the Web page document. When the styles are modified, the XHTML remains intact. This means that if your client decides to change something as small as the background color or as potentially huge as the page layout, you may only need to change one file that contains the styles, instead of each Web page document. For a look at how very powerful this can be, visit http://www.csszengarden.com and be amazed at how different pages with the same content and XHTML code (but different CSS) can look!
- **Potentially Smaller Documents.** Since both the formatting and page layout are separate from the document, the actual .html documents should be smaller.
- **Easier Site Maintenance.** Again, if the styles or page layout need to be changed it may be possible to complete the modifications by changing a single file only—the style sheet.
- **Increased Page Layout Control.** CSS used in conjunction with modern standards-compliant browsers provides a variety of positioning options (even down to the pixel) along with an ability to overlap elements. This gives the Web developer more control over the layout compared to the use of the previously popular XHTML tables.
- **Increased Accessibility.** Pages designed using XHTML tables for layout are easy to view with a traditional browser but can be very tedious when using a screen reader or other assistive technology. By reserving the use of XHTML tables for organizing tabular information and using CSS for page layout—the pages become more accessible.
- **Ability to Define Styles for Multiple Media Types.** Since presentation is separated from content, CSS can be used to set a separate style for printing, or possible use of a screen reader.
- **Support of the Semantic Web.** The Semantic Web is Tim Berners-Lee's vision of the future of the Internet (http://www.w3.org/2001/sw/). According to Berners-Lee, "The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." While much development is being done in this area, Web developers can take small steps, including using XHTML syntax and using CSS to separate styles from structure.

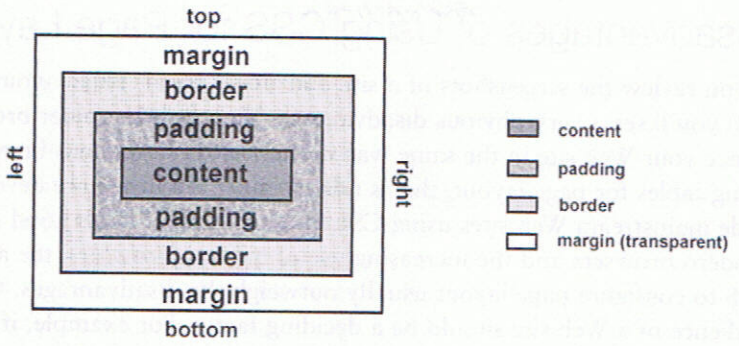


Figure 6.4
The CSS box model

- The **content** area can consist of a combination of text and Web page elements such as images, paragraphs, headings, lists, and so on. The **visible width** of the element on a Web page is the total of the content width, the padding width, and the border width.
- The **padding** area is between the content and the border. The default padding value is zero. When configuring the background of an element, the background is applied to both the padding and the content areas.
- The **border** area is between the padding and the margin. The default border has a value of 0 and does not display. You have already worked with borders in Hands-On Practice 4.2. As shown in Figure 4.2, this area can be set to various styles.
- The **margin** determines the empty space between the element and any adjacent elements. The margin is always transparent. The solid line in Figure 6.4 that contains the margin area does not display on a Web page. Keep in mind that browsers often have default margin values set for the Web page document and for certain elements such as paragraphs, headings, forms, and so on. Use the `margin` property to override the default browser values.

Figures 6.5 and 6.6 display `<div>` elements containing text content. Let's take a closer look. Figure 6.5 shows a screenshot of two `<div>` elements placed one after another on a Web page. In Figure 6.6 the boxes are nested inside each other. In both cases, the

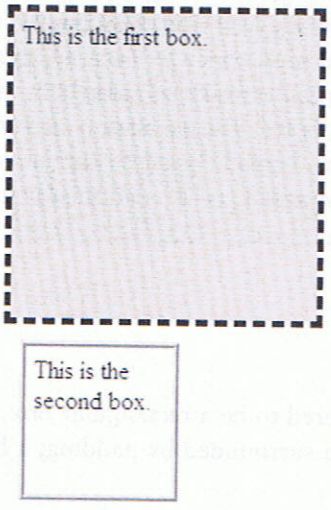


Figure 6.5
Two `<div>` elements displaying the box model

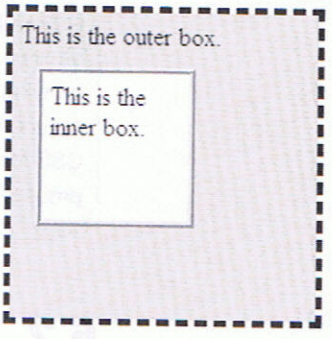


Figure 6.6
Nested elements showing the box model

Practice with Normal Flow and Nested Elements

Launch a text editor and open the box1.html file from the student files. Save the file with the name box2.html. This page is displayed in Figure 6.6.

Edit the code. You will not modify the CSS but you will edit the XHTML. Delete the content from the body section of the Web page. Add the following code to configure two `<div>` elements— one nested inside the other.

```
<div class="div1">
This is the outer box.
  <div class="div2">
  This is the inner box.
  </div>
</div>
```

Save the file. Launch a browser and test your page. It should look similar to the one shown in Figure 6.6. Notice how the browser renders the nested `<div>` elements— this is an example of normal flow. The student files contain a sample solution at Chapter6/box2.html.

The examples in the Hands-On Practice happened to use two `<divs>`. However, the box model applies to XHTML elements in general—not just to `<divs>`. You will get more practice using the box model in this chapter. Notice that since the CSS did not use any configurations for positioning, normal flow was used and the second box is nested within the first box because it is coded inside the first `<div>` in the XHTML. Next, we will take a look at some properties that affect positioning: `position`, `float`, `display`, and `z-index`.

6.3 CSS Positioning Properties

You've seen how normal flow causes the browser to render the elements in the order that they appear in the XHTML source code. When using CSS for page layout there are times where you will want to specify the location of an element on the page—either the absolute pixel location, the location relative to where the element would normally display, or floating on the page. There are even times when you will want to modify the way an element displays or cause an element to appear partially or completely over another element. The CSS properties used to accomplish these tasks are discussed next.

Relative and Absolute Positioning

Use **relative positioning** to change the location of an element slightly, relative to where it would otherwise appear in normal flow. Use the `position:relative` property along with either a left, right, and/or top property. The **`left` property** configures the position of the element in relation to the left side of the browser window. The **`right` property** sets the position of the element in relation to the right side of the browser window. The **`top` property** indicates the position of the element in relation to the top of the document area in the browser window.

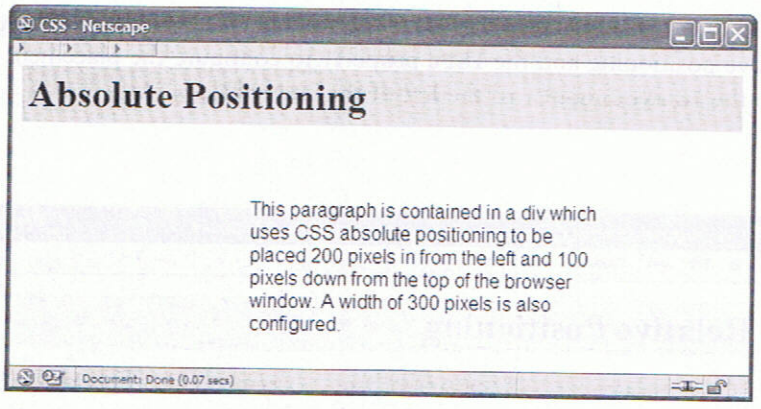


Figure 6.8
The paragraph is configured using absolute positioning

```
#content { position: absolute;
           left: 200px;
           top: 100px;
           font-family: Arial,sans-serif;
           width: 300px;
}
h1 { background-color: #cccccc;
     padding: 5px;
     color: #000000;
}
```

The XHTML source code follows:

```
<h1>Absolute Positioning</h1>
<div id="content">
<p>This paragraph is contained in a div which uses CSS absolute
positioning to be placed 200 pixels in from the left and 100 pixels
down from the top of the browser window. A width of 300 pixels is
also configured.</p>
</div>
```

When working with absolute positioning it is important to be aware that elements not absolutely positioned will be rendered following normal flow by the browser. Elements that are absolutely positioned are rendered outside of normal flow. You'll explore this behavior in the next Hands-On Practice.

HANDS-ON PRACTICE 6.2

Figure 6.9 shows screenshots of two Web pages with similar XHTML content. The Web page in the upper screenshot does not have any CSS applied. The Web page in the lower screenshot uses CSS to configure text, color, and the absolute position of a paragraph element. Launch a text editor and open the Chapter6/starter2.html found in the student files. When a browser renders the page it will use normal flow and display the XHTML elements in the same order as they are coded: `<h1>`, `<div>`, `<p>`, and `<ul>`. Launch a browser and display the page to verify.

4. Assign the paragraph to the `content` class. Add `class="content"` to the opening paragraph tag in the body of the Web page.

Save the file. Launch a browser and test your page. It should look similar to the lower Web page shown in Figure 6.9. The student files contain a sample solution at Chapter6/trillium.html. Note that even though the unordered list is coded in the page after the paragraph, it's displayed immediately after the heading. This is because the paragraph is absolutely positioned (`position: absolute`). Browsers render absolutely positioned elements *outside* of normal flow.

Note: This Hands-On Practice used embedded CSS for ease of editing. However, for an actual Web site with more than one page the most efficient solution is to use an external CSS file. See Chapter 3 if you'd like to review using external style sheets. You'll use external style sheets later in this chapter.

FAQ **What's a good name for a class?**

A class name should be descriptive of the purpose (such as `nav`, `news`, `footer`, and so on) rather than being descriptive of the presentation (such as `redText`). According to Google's Web Authoring Statistics Study, http://code.google.com/webstats, the 10 most commonly used class names are `footer`, `menu`, `title`, `small`, `text`, `content`, `header`, `nav`, `copyright`, and `button`.

The `float` Property

Elements that seem to float on the right or left side of either the browser window or another element are often configured using the **`float` property**. The browser renders these elements using normal flow, and then shifts them as far as possible within their container (usually either the browser window or a `<div>`) to either the right or left. Other content will flow around the float. To stop this flow, use the **`clear` property**. When floating an image, the `margin` property is useful to configure empty space between the image and text on the page.

Figure 6.10 shows a Web page (see the student files, Chapter6/float.html) with an image that has been configured with `float:right`.

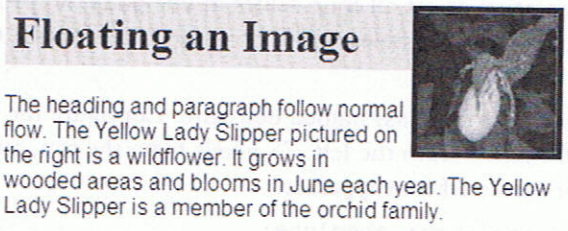


Figure 6.10
The image is configured with `float:right`

Notice how the text in the paragraph wraps around the image. An `id` called `yls` was created that applies the `float`, `margin`, and `border` properties. The attribute `id="yls"` was placed on the image tag. The CSS follows:

The XHTML source code follows:

```
<img class="rightfloat" src="yls.jpg" alt="Yellow Lady Slipper"
height="100" width="100" />
<p>The paragraphs 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>
<img class="rightfloat" src="pls.jpg" alt="Showy Lady Slipper"
height="100" width="100" />
<p>The paragraphs follow normal flow. The Showy Lady Slipper pictured
on the right is a wildflower. It is quite rare. It grows at the edge
of woodlands and blooms in July each year.</p>
<img class="rightfloat" src="trillium.jpg" alt="Trillium" height="100"
width="100" />
<p>The paragraphs follow normal flow. The Trillium pictured on the
right is a wildflower. It grows in shady woodlands and blooms in May
each year.</p>
```

The `float` and `clear` properties will also be useful when configuring page layouts with multiple columns. In the next Hands-On Practice you'll gain experience using the CSS `float` property.

FAQ

How do I know when to configure a `class` or an `id`?

If the style can be used on more than one element on a page, configure the style as a `class`. Use the `.` (dot) notation in the style sheet. Use the `class` attribute in the XHTML.

If the style is specific to only one element or if the element will be manipulated using DHTML (see Chapter 11), configure the style as an `id`. Use the `#` notation in the style sheet. Use the `id` attribute in the XHTML.

HANDS-ON PRACTICE 6.3

In this Hands-On Practice you'll practice using the CSS `float` and `clear` properties as you configure the Web page shown in Figure 6.12.

Create a folder named ch6float. Copy the starter3.html and yls.jpg files from the chapter6 folder in the student files into your ch6float folder. Launch a text editor and open the starter3.html file. Notice the order of the images and paragraphs. Notice that there is no CSS configuration for floating the images. Display starter3.html in a browser. The browser renders the page using normal flow and displays the XHTML elements in the order they are coded.

Let's add CSS to float the images and look more similar to Figure 6.12. Save the file with the name floatyls.html. With floatyls.html open in a text editor, modify the code as follows:

1. Add a style rule for a class name `float` that configures `float`, `margin`, and `border` properties.

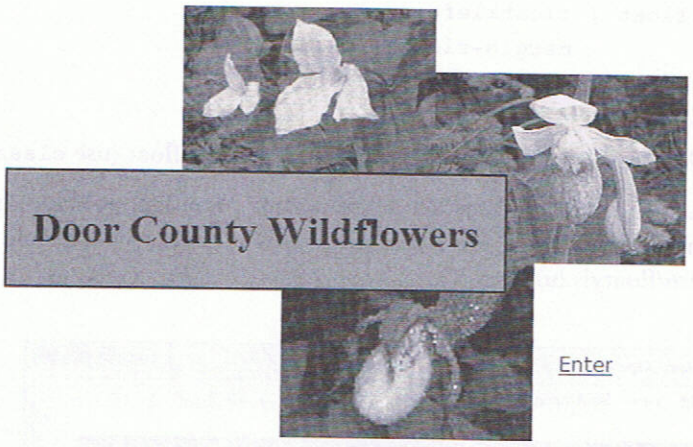


Figure 6.13
This page uses absolute positioning and `z-index` properties

Notice how the three flower photos and the logo are arranged. It would be difficult to recreate this just using XHTML. This type of page design may be appropriate for the splash page of a Web site. You will recreate this Web page when you complete the next Hands-On Practice . The term splash page originates from client-server applications that display an introductory (or splash) screen while the program loads. Splash pages, sometimes called splash screens, can set the tone or introduce a Web site.

You have been introduced to the `position`, `float`, `display`, and `z-index` properties. For your reference, Table 6.1 contains a list of CSS properties often used with formatting and page layout.

Table 6.1 CSS properties used with formatting and page layout

Property	Description	Commonly Used Values
`background-color`	Background color on an element	Any valid color
`background-image`	Background image on an element	`url(imagename.gif)` or `url(imagename.jpg)`
`background-position`	Position of the background image	Two percentage values or numeric pixel values. The first value configures the horizontal position and the second configures the vertical position starting from the upper-left corner of the container's box. Text values can also be used: `left`, `top`, `center`, `bottom`, `right`.
`background-repeat`	Controls how the background image will repeat	Text values `repeat` (default), `repeat-y` (vertical repeat), `repeat-x` (horizontal repeat), `no-repeat` (no repeat)
`border`	Shorthand notation to configure the `border-width`, `border-style`, and `border-color` of an element	The values for `border-width`, `border-style`, and `border-color` separated by spaces. For example: `border:1px solid #000000;`
`border-color`	Color of the border around an element	Any valid color
`border-style`	Type of border around an element	Text values `double`, `groove`, `inset`, `none` (the default), `outset`, `ridge`, `solid`, `dashed`, `dotted`, `hidden`

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

Property	Description	Commonly Used Values
`margin-bottom`	Size of an element's bottom margin	A numeric value (`px` or `em`) or percentage
`margin-left`	Size of an element's left margin	A numeric value (`px` or `em`) or percentage
`margin-right`	Size of an element's right margin	A numeric value (`px` or `em`) or percentage
`margin-top`	Size of an element's top margin	A numeric value (`px` or `em`) or percentage
`min-width`	The minimum width of an element	A numeric value (`px` or `em`) or percentage
`overflow`	Controls the display of a block-level element if the element exceeds its set height or width	Text values `visible`, `hidden`, `auto`, `scroll`
`padding`	Shorthand notation to configure the amount of padding—the blank space between the element and its border	Two numeric values (`px` or `em`) or percentages. The first value configures the top and bottom padding, the second value configures the left and right padding: `padding: 20px 10px;` Four numeric values (`px` or `em`) or percentages. The values configure the padding in the following order: `padding-top`, `padding-right`, `padding-bottom`, `padding-left`.
`padding-bottom`	Blank space between an element and its bottom border	A numeric value (`px` or `em`) or percentage
`padding-left`	Blank space between an element and its left border	A numeric value (`px` or `em`) or percentage
`padding-right`	Blank space between an element and its right border	A numeric value (`px` or `em`) or percentage
`padding-top`	Blank space between an element and its top border	A numeric value (`px` or `em`) or percentage
`position`	Configures the positioning of an element	The value `relative` will position the element in relation to the normal flow. The value `absolute` will position the element at the exact pixel location.
`right`	Distance in from the right to display an element	A numeric pixel value or percentage
`scrollbar-arrow-color`	Color of the arrow on the scroll bar (IE only)	Any valid color
`scrollbar-face-color`	Color of the sliding scroll bar (IE only)	Any valid color
`scrollbar-track-color`	Color of the track the scroll bar slides in (IE only)	Any valid color
`text-align`	The alignment of text	Text values `center`, `justify`, `left`, `right`
`text-decoration`	Determines whether text is underlined; this style is most often applied to hyperlinks	The text value `none` will cause a hyperlink not to be underlined in a browser that normally processes in this manner. The text value `underline` will configure hyperlink to be underlined.

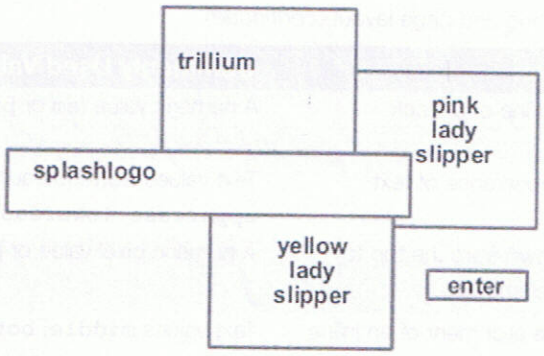


Figure 6.14
A sketch of the splash page elements

```
<?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>Door County Wildflowers</title>
  <link rel="stylesheet" href="wildflower.css" type="text/css" />
</head>
<body>
<div id="splashlogo">
  <h1>Door County Wildflowers</h1>
</div>
<div id="trillium">
  <img src="trillium.jpg" alt="Trillium" width="200" height="150" />
</div>
<div id="yls">
  <img src="yls.jpg" alt="Yellow Lady Slipper" width="200" height="150" />
</div>
<div id="pls">
  <img src="pls.jpg" alt="Pink Lady Slipper" width="200" height="150" />
</div>
<div id="enter">
  <a href="page1.html">Enter</a>
</div>
</body>
</html>
```

Save your file as splash.html in the wildflowers folder. Test your page in a browser and compare it with the one shown in Figure 6.15.

The browser has displayed the page using normal flow—following the order of the elements in the source code. Don't worry that your elements are not positioned as they should be. Next, you'll configure the positioning properties as you create the wildflower.css.

Part 2—Code the External Style Sheet

Let's take a moment to consider what type of positioning is needed for the splash page: the splash screen logo, the three images, and the link. Refer to the wireframe sketch in

- **Yellow Lady Slipper Image Area**

```
#yls { position: absolute;
       z-index: 3;
       left: 300px;
       top: 270px;
}
```

- **Hyperlink Area**

```
#enter { position: absolute;
         left: 520px;
         top: 350px;
         font-family: Verdana,sans-serif;
}
```

Save the wildflower.css file in the wildflowers folder.

How do I know exactly what values to use for absolute positioning?

Lots of testing! When hand-coding a page that uses absolute positioning there is a lot of trial and error. It is helpful to sketch the page or create a prototype using a graphics application. Be patient—make your best guess, test, and repeat until the configuration is complete. The good news is that Web authoring tools such as Adobe Dreamweaver provide a visual editor that greatly streamlines this process.

Part 3—Test the Splash Page

Now that your styles are coded, test the splash.html page again. Your page should be similar to the screenshot shown in Figure 6.13. If there are differences, verify the `id` 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 wildflower.css and splash.html in the Chapter6 folder.

CHECKPOINT 6.1

1. State three reasons to use CSS for page layout on a commercial site being developed today.
2. Describe the difference between relative and absolute positioning.
3. Describe the purpose of the `z-index` CSS property.

6.4 Exploring CSS Page Layout

You've just configured a Web page using CSS to position elements. You'll continue to explore using CSS for page layout as you create another page for the Door County Wildflowers site. As discussed earlier in this chapter, the CSS `float` property causes the browser to display XHTML elements outside of the normal flow. Elements configured

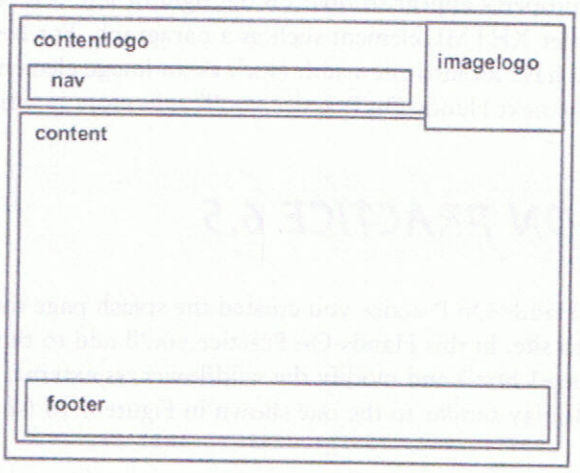


Figure 6.17
Note how the elements are arranged on the page

```
<?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>Door County Wildflowers</title>
  <link rel="stylesheet" href="wildflower.css" type="text/css" />
</head>
<body>
<img id="imagelogo" src="pls.jpg" alt="Pink Lady Slipper"
width="200" height="150" />
<div id="contentlogo">
  <h1>Door County Wildflowers</h1>
  <a class="nav" href="home.html">Home</a>
  <a class="nav" href="spring.html">Spring</a>
  <a class="nav" href="summer.html">Summer</a>
  <a class="nav" href="fall.html">Fall</a>
  <a class="nav" href="winter.html">Winter</a>
</div>
<div class="content">
  <p>Wisconsin&#39;s Door County Peninsula is a unique, ecologically
diverse place with upland and boreal forest, bogs, swamps, sand and
rock beaches, limestone escarpments, and farmlands. A wide array of
wildflowers grow in the county because of this variety of
ecosystems.</p>
  <p>Explore the beauty of Door County Wildflowers....</p>
  <div class="footer">
    Copyright &copy; 2007 Door County Wild Flowers<br />
    Last Updated on 07/02/07
  </div>
</div>
</body>
</html>
```

```
#imagelogo { float: right;
             margin: 0 0 5px 5px;
             border: solid;
}
```

- **Navigation Links.** Configure the nav class to use 5 pixel padding, a background color of `#e8b9e8`, text color of dark blue (`#000066`), no underline (`text-decoration:none`), Verdana, Arial or other sans-serif font, and center-aligned text. The CSS follows:

```
.nav { padding: 5px;
       background-color: #e8b9e8;
       color: #000066;
       text-decoration: none;
       font-family: Verdana, Arial, sans-serif;
       text-align: center;
}
```

- **Overall Content.** Configure the content class to use Verdana, Arial, or sans-serif font and have a margin of 10 pixels.

```
.content { font-family: Verdana,Arial,sans-serif;
           margin: 10px;
}
```

- **Page Footer.** Configure a class called footer with `xx-small` font that is centered as follows:

```
.footer { font-size: xx-small;
          text-align: center;
}
```

Save the wildflower.css file in the wildflowers folder.

Part 3—Test the Content Page

Now that your styles are coded, test the page1.html page again. Your display should be similar to the screenshot shown in Figure 6.16. If there are differences, verify the `id` and `class` values in your XHTML. Also 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 splash.html, page1.html, and wildflower.css in the Chapter6/wildflowers folder.

6.5 Two-Column Page Layout

A common design for a Web page is a two-column layout with left-column navigation and right-column logo and content. Figure 6.19 shows a page designed in this format using CSS.

The page contains a number of elements, as shown in Figure 6.19. Compare the wireframe sketch in Figure 6.20 with the page displayed in Figure 6.19 (shown also in the color insert section).

ating a series of nested boxes. Refer to the Web page screenshot shown in Figure 6.19 and the page layout sketch in Figure 6.20 as you complete this Hands-On Practice.

Getting Started

Locate the showy.jpg file in the Chapter6 folder in the student files. Create a new folder called wildflowers2. Copy the file to the folder.

Part 1—Code the XHTML

Review Figures 6.19 and 6.20. Notice the following page elements: wrapper, left column, right column, logo, navigation bar, content, right-floating image, and footer. Each of these areas will be coded to use an `id` and `class` that corresponds to CSS, which configures a number of properties including the `padding`, `border`, `font-family`, and so on. As you code the XHTML document, you will place the elements on the page, assigning `id` and `class` values that correspond to the areas in the sketch shown in Figure 6.20. Launch Notepad and type in the following XHTML:

```
<?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>Door County Wildflowers</title>
</head>
<body>
<div id="wrapper">
  <div id="leftcolumn">
    <a class="navBar" href="index.html">Home</a>
    <a class="navBar" href="spring.html">Spring</a>
    <a class="navBar" href="summer.html">Summer</a>
    <a class="navBar" href="fall.html">Fall</a>
    <a class="navBar" href="winter.html">Winter</a>
  </div>
  <div id="rightcolumn">
    <div id="logo">
      <h1>Door County Wildflowers</h1>
    </div>
    <div class="content">
      <img src="showy.jpg" width="200" height="150" id="floatright"
        alt="Showy Lady Slippers" />
      <p>Wisconsin&#39;s Door County Peninsula is a unique,
ecologically diverse place with upland and boreal forest, bogs,
swamps, sand and rock beaches, limestone escarpments, and farmlands.
</p>
      <p>A wide array of wildflowers grow in the county because of
this variety of ecosystems.</p>
      <h3>Explore the beauty <br />of Door County Wildflowers</h3>
    </div>
    <div class="footer">
```

Now let's consider the CSS configuration. Type the CSS in your document as it is discussed as follows:

- **Body Tag.** This should be configured with the default fonts of Verdana, Arial, or any sans-serif font. The page margin should be set to 0 pixels.

```
body { font-family: Verdana,Arial,sans-serif;
       margin: 0;
}
```

- **Wrapper.** Configure a container, or wrapper, area to contain the two columns and configure default background (`#e8b9e8`) and text (`#000066`) colors.

```
#wrapper { background-color: #e8b9e8;
           color: #000066;
}
```

- **Left Column.** The key to this two-column page layout is that the left 100 pixel wide column is designed to float to the left. The left column uses the background and text colors configured in the wrapper id.

```
#leftcolumn { float: left;
              width: 100px;
}
```

- **Right Column.** Since the left column is 100 pixels wide and floats on the left side, assign a 100 pixel wide left margin to the right column. Configure a background (`#ffffff`) and text (`#000000`) color.

```
#rightcolumn { margin-left: 100px;
               background-color: #ffffff;
               color: #000000;
}
```

- **Logo.** The logo is configured with a background color of `#eeeeee`, text color of `#cc66cc`, an extra large font size, 10 pixels of padding, and a solid black border that is 1 pixel wide.

```
#logo { background-color: #eeeeee;
        color: #cc66cc;
        font-size: x-large;
        border-bottom: 1px solid #000000;
        padding: 10px
}
```

- **Content.** The content area will be easier to read if there is additional empty space on the top, right, and left sides. This can be configured using the `padding` property. Typically, you have set the same `padding` value for all four sides of the element's box. The padding set for the logo is an example of this. Padding can be set for each side individually using the `padding-top`, `padding-right`, `padding-bottom`, and `padding-left` properties. A short-hand version of this is 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). In this page layout the padding at the top, right, and left sides of the content should be set to 20 pixels.

```
.content { padding: 20px 20px 0 20px;
}
```

FAQ

How do I create a custom-color scroll bar?

It can be fun to color-coordinate the scroll bar with your Web site! Keep in mind that not all your Web visitors will see your handiwork. While this effect is supported by Internet Explorer, it is not supported by all browsers. To configure a scroll bar with colors that you choose, add the following styles to the body tag: `scrollbar-face-color`, `scrollbar-arrow-color`, `scrollbar-track-color`. For example:

```
body {  scrollbar-face-color: #cc66cc;
        scrollbar-arrow-color: #006600;
        scrollbar-track-color: #cccccc;
}
```

Note: Your CSS will not pass W3C validation tests if you use these Internet Explorer only properties.

6.6 CSS Debugging Tips

Using CSS for page layout requires some patience. It takes a while to get used to it. One of the biggest issues is that even modern browsers implement CSS in slightly different ways. Testing is crucial. Don't make it your goal that the pages must look exactly the same on every browser. Expect your pages to look slightly different on various browsers. Design so they look best on the most commonly used browser (currently Internet Explorer) and display acceptably well on other browsers. There are Web pages devoted to CSS bugs and browser support of CSS. The following are a few that you will find helpful:

- http://web.archive.org//20040202153928/http://devedge.netscape.com/library/xref/2003/css-support/css1/mastergrid.html

  The original "Master List" created by Eric Meyer
- http://www.westciv.com/style_master/academy/browser_support/index.html

  A comprehensive browser compatibility list
- http://www.positioniseverything.net

  John and Holly Bergevin's site focuses on CSS bugs in modern browsers—it contains some great sample CSS page layouts
- http://www.quirksmode.org

  Peter-Paul Koch's site is dedicated to studying and defeating browser incompatibilities related to CSS and JavaScript

CSS Debugging Techniques

Debugging CSS can be frustrating. The following are helpful techniques to use:

- **Manually Check Syntax Errors.** Sometimes a CSS style does not apply due to a syntax error. Carefully check your code. Many times the error is in the line above the style that is not correctly applied.
- **Programmatically Check Syntax Errors.** As mentioned earlier, you can use the W3C's CSS Validator at http://jigsaw.w3.org/css-validator to verify your syntax.

- http://www.w3.org/Style/CSS/learning
  W3C's list of CSS resources
- http://www.bluerobot.com/web/layouts
  A "reservoir" of CSS page layouts
- http://www.blooberry.com/indexdot/css
  CSS syntax reference list
- http://www.w3.org/TR/1998/REC-CSS2-19980512
  W3C CSS Level 2 Recommendation
- http://www.w3.org/TR/REC-CSS1-961217.html
  W3C CSS Level 1 Recommendation

CHECKPOINT 6.2

1. The two-column layouts you created in the previous Hands-On Practice did not use absolute positioning. Open the twocolumn.html page in a browser. Resize the browser window. Describe what happens. What type of page design layout (ice, jello, or liquid) is being used?
2. Describe one CSS debugging tip that you have found helpful.
3. Describe how to choose whether to configure an XHTML tag, create a class, or create an id when working with CSS.

7. Which of the following will configure padding that is 10 pixels on the top, 0 pixels on the left and right, and 5 pixels on the bottom?

a. `margin: 0px 5px 0px 10px;`
b. `margin:top-10, left-0, right-0, bottom-5;`
c. `padding: 10px 0 5px 0;`
d. none of the above

8. Which of the following is used along with the `left`, `right` and/or `top` property to configure the position of an element precisely?

a. `position: relative`
b. `position: absolute`
c. `position: float`
d. none of the above

9. Which of the following configures a margin for an element with the following values: top margin 20 pixels, left margin 300 pixels, right margin 0 pixels, and bottom margin 0 pixels?

a. `margin: 300px 20px 0px 300px;`
b. `margin: top-20, left-300, right-0, bottom-0;`
c. `margin: 20px 0 0 300px;`
d. none of the above

10. Which of the following will configure a class called `news` to stack on top of other elements that have a `z-index` of 5?

a. `.news { z-index: high} ;`
b. `.news { z-index:6} ;`
c. `.news { z-index:4} ;`
d. `none of the above`

Fill in the Blank

11. Configure a style with a(n) _____ if the style will only apply to one element on a page.

12. If an element is configured with `float:right`, the other content on the page will appear to its _____.

13. The _____ is always transparent.

14. Use the `position:relative` property along with the _____, _____, and/or _____ property to configure the position of an element in relation to the normal flow.

15. Configure a style with a_____ if the style could apply to more than one element on a page.

Apply Your Knowledge

1. **Predict the Result.** Draw and write a brief description of the Web page that will be created with the following XHTML code:

```
<html>
<head>
  <title>CircleSoft Web Design</title>
  <style type="text/css">
  h1 { border-bottom: 1px groove #333333;
       color: #006600;
       background-color: #cccccc
  }
  #content { position: absolute;
             left: 200px;
             top: 75px;
             font-family: Arial,sans-serif;
             width: 300px;
  }
  .nav { font-weight: bold;
  }
  </style>
</head>
```

```
.footer { font-size: xx-small;
          text-align: center;
          clear: "_";
}
.navBar { color: #000066;
          text-decoration: none;
          padding: 3px;
          margin: 15px;
          display: "_";
}
</style>
</head>
<body>
<div id="rightcolumn">
  <a class="navBar" href="index.html">Home</a>
  <a class="navBar" href="products.html">Products</a>
  <a class="navBar" href="services.html">Services</a>
  <a class="navBar" href="about.html">About</a>
</div>
<div id="leftcolumn">
  <div id="logo">
    <h1>Trillium Media Design</h1>
  </div>
  <div class="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>
  <div class="footer">
    Copyright &copy; 2008 Trillium Media Design<br />
    Last Updated on 01/15/08
  </div>
</div>
</body>
</html>
```

3. Find the Error. When this page is displayed using Internet Explorer 7, the heading information obscures the paragraph text. Correct the errors and describe the process you followed.

```
<?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>CSS Float</title>
<style type="text/css">
```

Code an external style sheet file called rightcolumn.css and a Web page called rightcolumn.html. The Web page should have two columns. The right column will be the navigation column and the left column will be the content column. Hand in printouts of rightcolumn.css, the rightcolumn.html source code (print in Notepad), and the browser display of your rightcolumn.html to your instructor.

7. **Extending Hands-On Practice 6.6.** Design a two-column page layout with a logo area across the top. Review the Hands-On Practice 6.6 twocolumn.html file for some examples. These files are in the Chapter6/wildflowers2 folder in the student files. Code an external style sheet file called mydesign.css and a Web page called mydesign.html. The Web page should have two columns and a logo area across the top. Hand in printouts of mydesign.css, the mydesign.html source code (print in Notepad), and the browser display of your mydesign.html to your instructor.

8. **Extending Hands-On Practice 6.6.** In Hands-On Practice 6.6 you created two files for a version of the Door County Wildflowers Web site. The files are available in the Chapter6/wildflowers2 folder in the student files. You will create two additional content pages for the Door County Wildflowers site, called spring.html and summer.html, in this exercise. Be sure that all CSS is placed in an external style sheet, called mywildflower.css. (Modify pre-existing pages to use this style sheet.) Here is some content to include on the new pages:

   Spring Page (spring.html):

   - Use the trillium.jpg image (see the Chapter6 folder in the student files).
   - Trillium facts: 8–18 inches tall, perennial, native plant, grows in rich moist deciduous woodlands, white flowers turn pink with age, fruit is a single red berry, protected flower species.

   Summer Page (summer.html):

   - Use the yls.jpg image (see the Chapter6 folder in the student files).
   - Yellow Lady's Slipper facts: 4–24 inches tall, perennial, native plant, grows in wet shaded deciduous woods, swamps, and bogs, an orchid, official flower of Door County.

   Hand in printouts of mywildflower.css, spring.html source code (print in Notepad), summer.html source code, the browser display of spring.html, and the browser display of summer.html to your instructor.

9. **Extending Hands-On Practice 6.2.** Modify the twocolumn.html page you created in Hands-On Practice 6.2. This file is in the Chapter6/wildflowers2 folder in the student files. Recall from Chapter 5 that a Web page using jello design has content in the center of the Web page with blank margins on either side. You can code this using CSS by configuring the margin property of the body tag to use percentages for the left and right. For example:

   ```
   body { margin: 0 10% 0 10%; }
   ```

   Hand in printouts of the source code (print in Notepad) and browser display for the Web page to your instructor.

10. Design a splash page called moviesplash.html about your favorite movie. Use absolute positioning and `z-index` to create an interesting display. First sketch the areas for images, text, and link to the first page on the site. Search the Web for

case study you will implement a new two-column CSS page layout for JavaJam. You will modify the external style sheet and the Home, Menu, Music, and Jobs pages. Unless your instructor directs you otherwise, use the Chapter 4 JavaJam Web site as a starting point for this case study.

Figure 6.22 displays a wireframe for the two-column page layout with a page container, logo, left column, navigation, right column, floating, and footer areas.

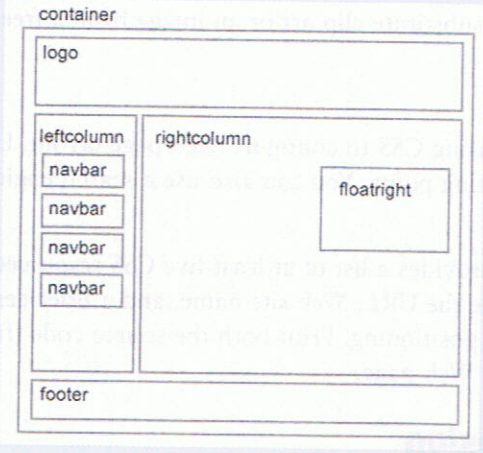


Figure 6.22
JavaJam two-column page layout

Hands-On Practice Case

1. **Create a Folder.** Create a folder called javajamcss. Copy all the files from your Chapter 4 javajam folder into the javajamcss folder. You will modify the javajam.css file and each Web page file (index.html, menu.html, and music.html) to implement the two-column page layout shown in Figure 6.22. See the new JavaJam Home page, as shown in Figure 6.23 (shown also in the color insert section).
2. **Configure the CSS.** Open javajam.css in Notepad. Edit the style rules as follows:
   - Modify the `container` id to have a minimum width of 700 pixels (use `min-width: 700px`), background (`#e8d882`) and text (`#000000`) color, 80% width, and a 2 pixel black double border (`border: 2px double #000000`).
   - Configure the logo area. Remove the `h1` selector and style rules. Create a new id named `logo` with a background (`#ccaa66`) and text (`#000000`) color, center alignment (`text-align: center`) and a bottom border that is 2 pixels, double, and black (`border-bottom: 2px double #000000`).
   - Configure the left column area. Add a new style rule for the `leftcolumn` id to configure an area that floats to the left, is 100 pixels wide, and has 10 pixels of padding on the top side.

```
#leftcolumn { float: left;
              width: 100px;
              padding-top: 10px;
}
```

   - Configure the navigation area. Remove the `nav` id. Add a new style rule for the `navbar` class to configure an area with no underlines on hyperlinks, a 15

- Modify the `footer` id to display a 2 pixel double black top border (`border-top: 2px double #000000`).

Save the javajam.css file.

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 JavaJam logo image. Code a `<div>` that surrounds the JavaJam 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 `navbar` class (`class="navbar"`).
   - Configure the right column. This area contains the remaining page content, including the footer area. Add a `<div>` that surrounds the text, winding road image, and footer area. Assign the `<div>` to the id `rightcolumn`.
   - Configure the area that floats to the right. Modify the winding road image element. Remove the `align="right"` attribute and add `class="floatright"` to the winding road image element.

   Save the index.html file. It should look similar to the Web page shown in Figure 6.23. Remember that validating your XHTML and CSS can help you find syntax errors. Test and correct this page before you continue.
4. **Modify the menu.html and music.html Files.** Modify the code in 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.
5. **Bonus Style: `text-transform`.** Figure 6.24 shows an alternate design for the music.html page. Notice how the `<h3>` elements are styled differently—with all uppercase text (using a new property, `text-transform`) different background and text colors, font size, bottom border, and margin. Open javajam.css in a text editor and replace the `h3` selector style rules with the following:

```
h3 { text-transform: uppercase;
     background-color: #ffffcc;
     color: #663300;
     font-size: 20px;
     border-bottom: 1px solid #000000;
     margin-right: 20px;
}
```

   Save the javajam.css file. Test your pages in a browser. Your music.html page should look similar to the one shown in Figure 6.23. The other pages do not use `<h3>` elements and should appear as they did at the end of Step 4.

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

Hands-On Practice Case

1. **Create a Folder.** Create a folder called fishcreekcss. Copy all the files from your Chapter 4 fishcreek folder into the fishcreekcss folder. You will modify the fishcreek.css file and each Web page file (index.html, services.html, and askvet.html) to implement the two-column page layout, as shown in Figure 6.25. See the new Fish Creek home page, as shown in Figure 6.26 (shown also in the color insert section).

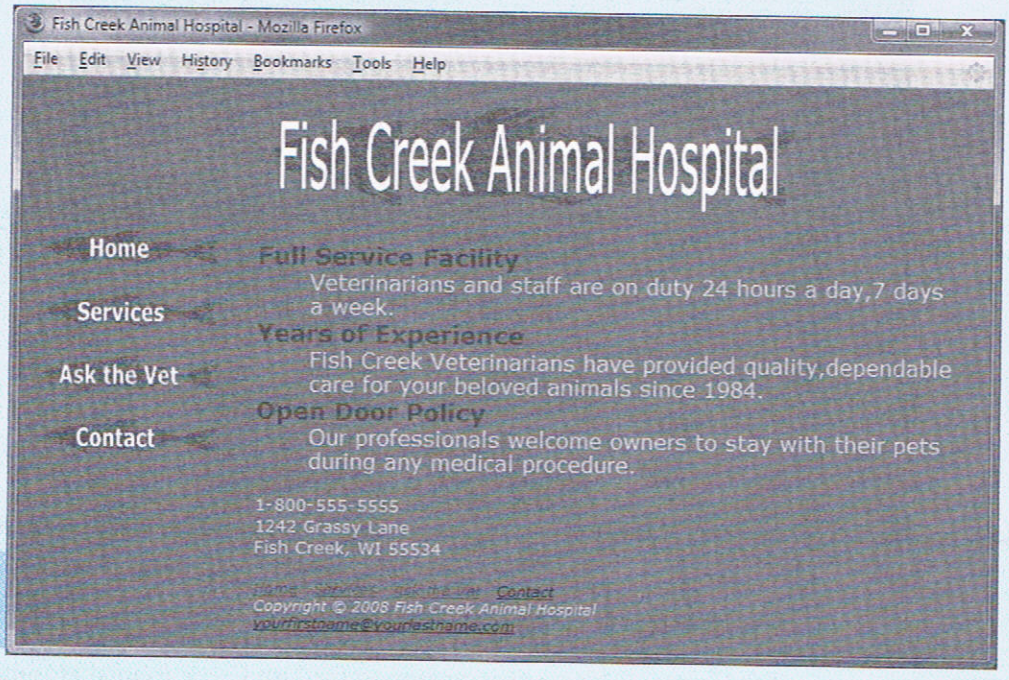


Figure 6.26
The new Fish Creek two-column index.html

See the center color insert

2. **Configure the CSS.** Open fishcreek.css in Notepad. Edit the style rules as follows:
   - Modify the container id style rules. Notice that the new page layout aligns with the left margin. Remove the properties previously used to center the content (`margin-left` and `margin-right`).
   - Configure the logo area. Remove the `h1` selector and style rules. Create a new id named `logo` with 70 pixels of padding on the left side (`padding-left: 70px`).
   - Configure the left column area. Add a new style rule for the `leftcolumn` id to configure an area that floats to the left and is 140 pixels wide.

```
#leftcolumn { float: left;
              width: 140px;
}
```

   - Configure the navigation area. Remove the `imgnav` id. Add a property to the style rules for the `img` selector—configure 10 pixels of padding (`padding: 10px`).
   - Configure the right column area. Add a new style rule for the `rightcolumn` id to configure an area with a 170 pixel left margin.
   - Configure the footer area. Remove the `nav` id.

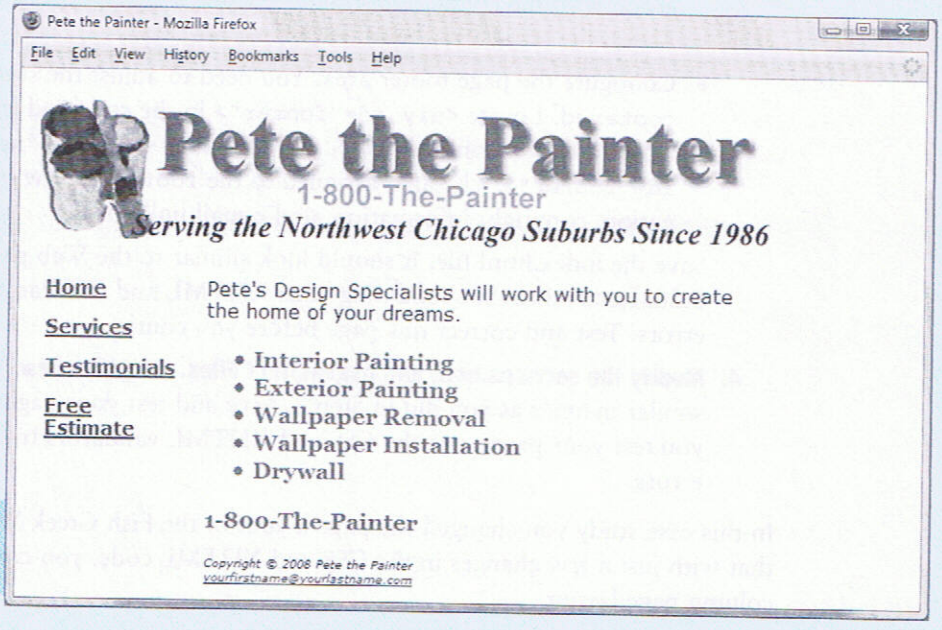
   Save the fishcreek.css file.

Hands-On Practice Case

1. **Create a Folder.** Create a folder called paintercss. Copy all the files from your Chapter 4 folder into the paintercss folder. You will modify the painter.css file and each Web page file (index.html, services.html, and testimonials.html) to implement the two-column page layout shown in Figure 6.27. See the new Pete the Painter home page, as shown in Figure 6.28 (shown also in the color insert section).

Figure 6.28
The new Pete the Painter two-column index.html

See the center color insert



2. **Configure the CSS.** Open painter.css in Notepad. Edit the style rules as follows:
   - Configure the left column area. Add a new style rule for the `leftcolumn` id to configure an area that floats to the left and is 150 pixels wide.

```
#leftcolumn { float: left;
              width: 150px;
}
```

   - Configure the navigation area. Remove the `nav` id. Add a new style rule for the `nav` class to configure an area with bold font, a 15 pixel margin on the right, bottom, and left sides, and is displayed as a block element (with line breaks above and below) by the browser.

```
.nav { font-weight: bold;
       margin: 0 15px 15px 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 and a 10 pixel top margin (`margin: 10px 0 0 150px`).
   - Configure an area that floats to the right. Add a new style rule for the `floatright` class.

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

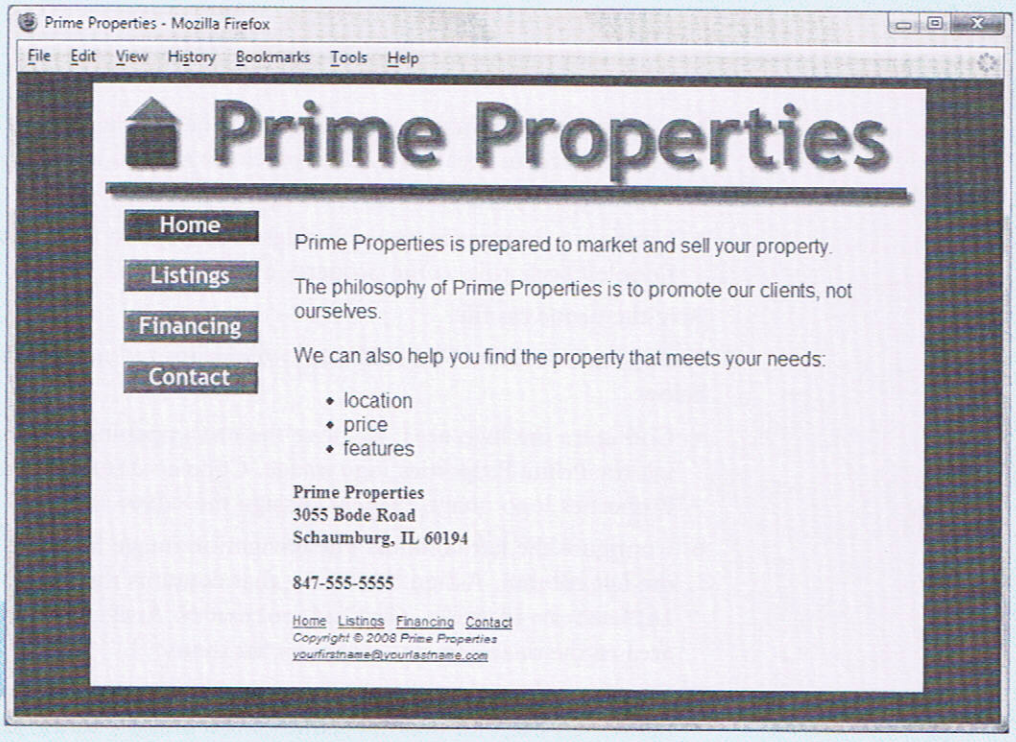
   Save the painter.css file.

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;
}
```

Web Project

See Chapter 5 for an introduction to the Web Project case. As you completed the Chapter 5 Web Project Case Study activities you completed a Web Project Topic Approval, Web Project Site Map, and Web Project Page Layout Design. In this case study you will use your design documents as a guide as you develop the pages for your Web Project using CSS (external style sheet) for both formatting and page layout.

Hands-On Practice Case

1. Create a folder called project. All your project files and graphics will be organized in this folder and subfolders as needed.
2. Refer to your Site Map to view the pages that you need to create. Jot down a list of the file names. Add these to the Site Map.
3. Refer to the Page Layout Design. Make a list of the common fonts and colors used on the pages. These may become the CSS you configure for the body element. Note where typical elements used for organization (such as headings, lists, paragraphs, and so on) may be used. You may want to configure CSS for these elements. Identify various page areas such as logo, navigation, footer, and so on—and list any special configurations needed for these areas. These will be configured as classes in your CSS. Create an external style sheet, called project.css, which contains these configurations.
4. Using your design documents as a guide, code a representative page for your site. Use CSS to format text, color, and layout. Be sure to apply classes and ids where appropriate. Associate the Web page to the external style sheet.

   Save and test the page. Modify both the Web page and the project.css file as needed. Test and modify until you have achieved the look you want.
5. Using the completed page as a template wherever possible, code the rest of the pages on your site. Test and modify them as needed.
6. Experiment with modifying the project.css file. Change the page background color, the font family, and so on. Test your pages in a browser. Notice how a change in a single file can affect multiple files when external style sheets are used.