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:

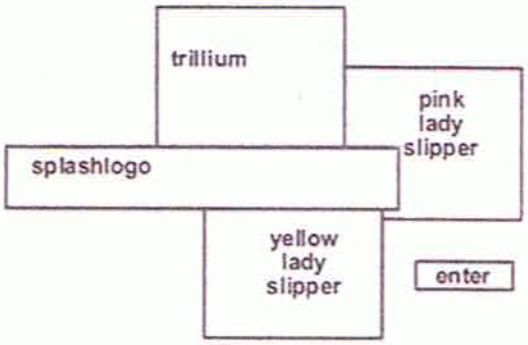

Figure 6.14
A sketch of the splash page elements

```
<?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

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;
}
```

- **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

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).

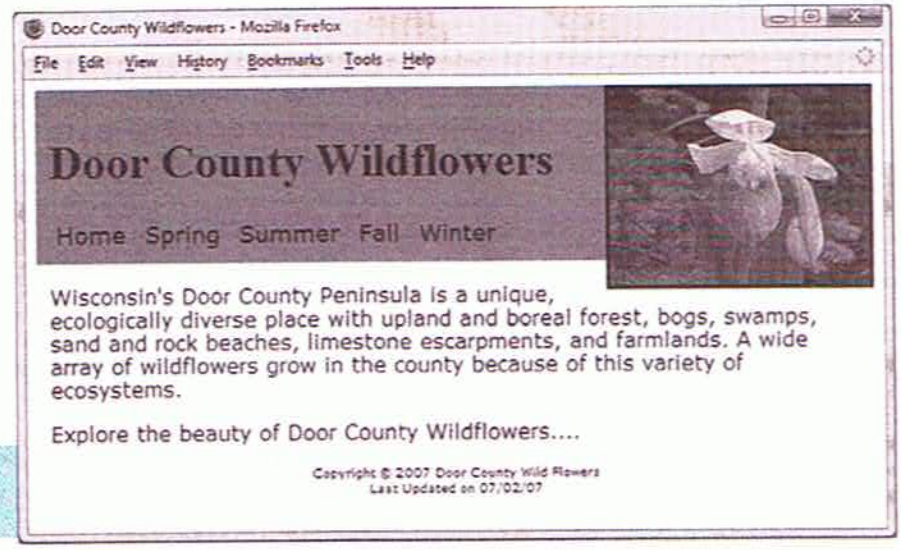

Figure 6.16
The image is floating on the page

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 `ids` and `classes` 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:

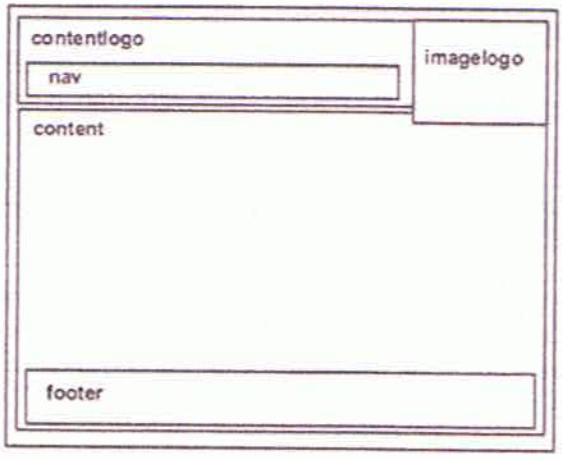

Figure 6.17
Note how the elements are arranged on the page

```
<?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
```

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.

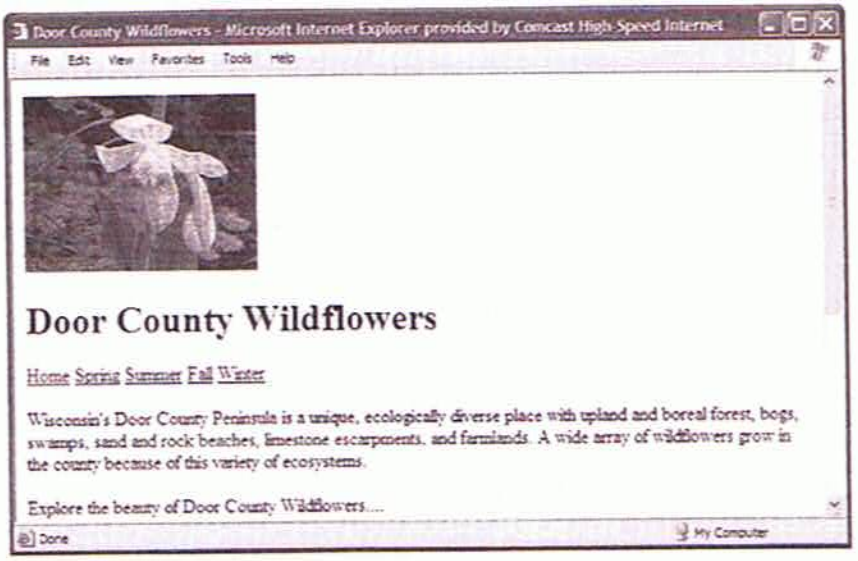

Figure 6.18
The page before CSS for positioning is configured

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:

```
#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).

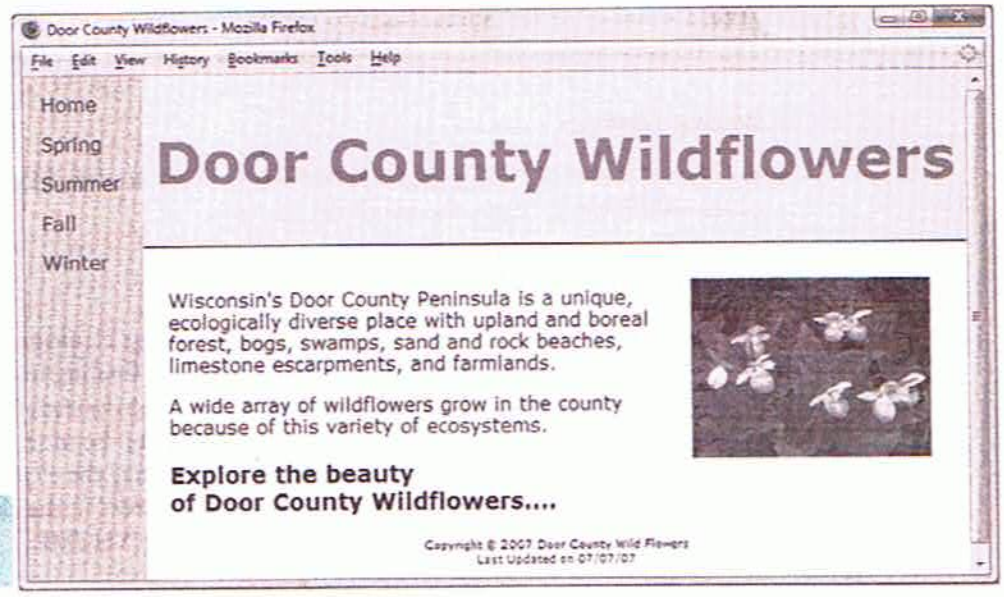

Figure 6.19
A two-column page configured using CSS

See the center color insert

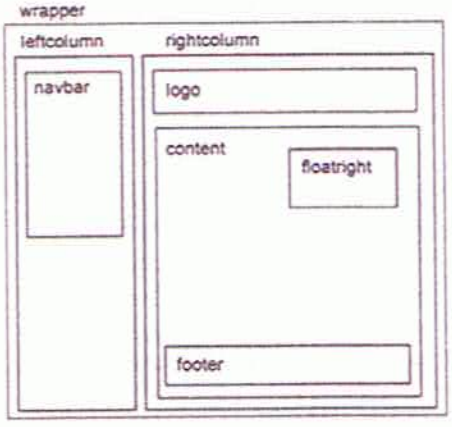

Figure 6.20
The two-column page layout

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-

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">
```

```
        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.

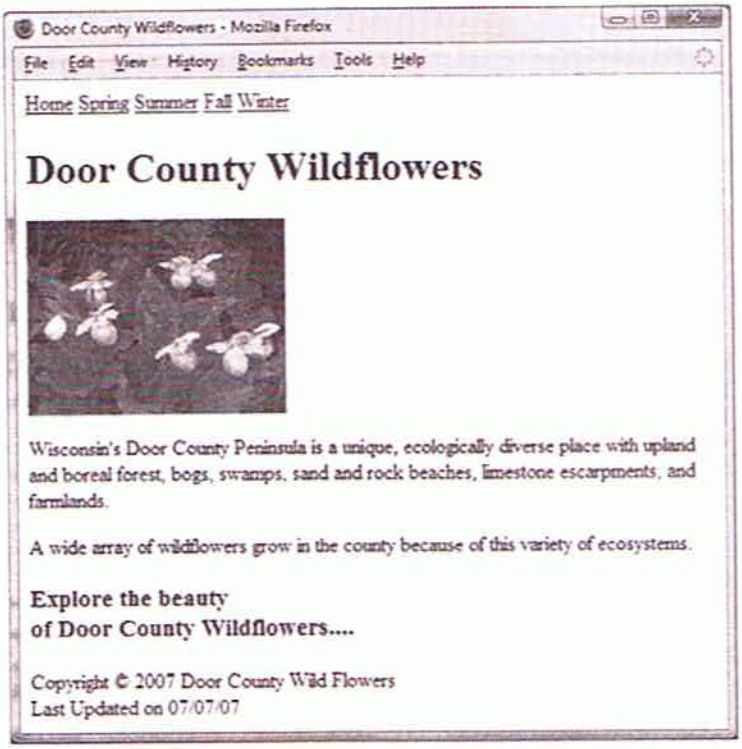

Figure 6.21
The two-column page before CSS is applied

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">`.

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;
}
```

- **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