6

Page Layout

Chapter Objectives

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

- Describe and apply the CSS Box Model
- Configure margins with CSS
- Configure float with CSS
- Configure positioning with CSS
- Create two-column page layouts using CSS
- Configure navigation in unordered lists and style with CSS
- Add interactivity to hyperlinks with CSS pseudo-classes
- Configure an interactive image gallery
- Configure web pages with HTML5 structural elements, including section, article, and aside
- Configure older browsers to be compatible with HTML5

You've already configured the centered page layout with CSS. We'll add to your toolbox of CSS page layout techniques in this chapter, starting with the box model. You'll explore floating and positioning elements with CSS. You'll be introduced to using CSS to add interactivity to hyperlinks with pseudo-classes and use CSS to style navigation in unordered lists. You will practice coding new HTML5 elements that structure web page content.

6.1 The Box Model

Each element in a document is considered to be a rectangular box. As shown in Figure 6.1, this box consists of a content area surrounded by padding, a border, and margins. This is known as the **box model**.

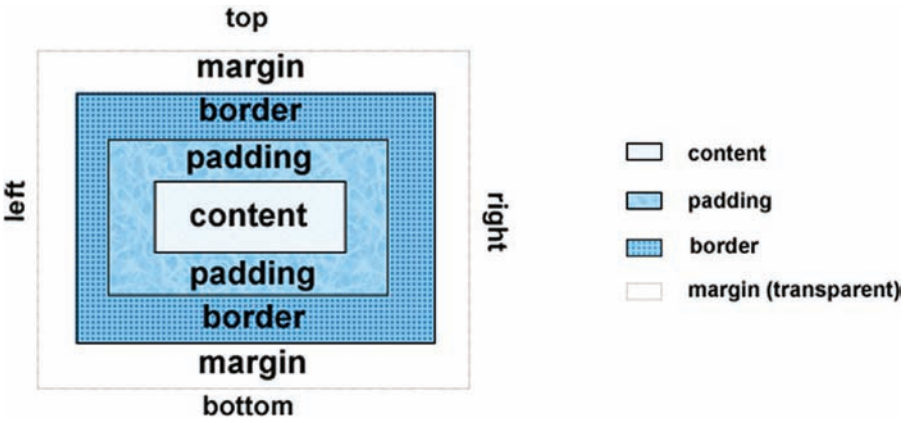

Figure 6.1 The CSS box model

Content

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. However, the **`width` property** only configures the actual width of the content; it does not include any padding, border, or margin.

Padding

The **padding** area is between the content and the border. The default padding value is zero. When the background of an element is configured, the background is applied to both the padding and the content areas. Use the `padding` property to configure an element's padding (refer to Chapter 4).

Border

The **border** area is between the padding and the margin. The default border has a value of `0` and does not display. Use the `border` property to configure an element's border (refer to Chapter 4).

Margin

The **margin** determines the empty space between the element and any adjacent elements. The solid line in Figure 6.1 that contains the margin area does not display on a web page.

The margin Property

Use the **`margin` property** to configure margins on all sides of an element. The margin is always transparent—the background color of the web page or parent element shows in this area. Browsers 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.

To configure the size of the margin, use a numeric value (px or em). To eliminate the margin, configure it to 0 (with no unit). Use the value `auto` to indicate that the browser should calculate the margin. In Chapters 3 and 4, you used `margin-left: auto;` and `margin-right: auto;` to configure a centered page layout. Table 6.1 shows CSS properties that configure margins.

Table 6.1 Configuring margins with CSS

Property	Description and Common Values
`margin`	Shorthand notation to configure the margin surrounding an element A numeric value (px or em) or percentage; for example, `margin: 10px;` The value `auto` causes the browser to automatically calculate the margin for the element Two numeric values (px or em) or percentages: The first value configures the top margin and bottom margin, and the second value configures the left margin and right margin; for example, `margin: 20px 10px;` Three numeric values (px or em) or percentages: The first value configures the top margin, the second value configures the left margin and right margin, and the third value configures the bottom margin; for example, `margin: 10px 20px 5px;` Four numeric values (px or em) or percentages; the values configure the margin in the following order: `margin-top`, `margin-right`, `margin-bottom`, and `margin-left`; for example, `margin: 10px 30px 20px 5px;`
`margin-bottom`	Bottom margin; a numeric value (px or em), percentage, or `auto`
`margin-left`	Left margin; a numeric value (px or em), percentage, or `auto`
`margin-right`	Right margin; a numeric value (px or em), percentage, or `auto`
`margin-top`	Top margin; a numeric value (px or em), percentage, or `auto`

The Box Model in Action

The web page shown in Figure 6.2 (chapter6/box.html in the student files) depicts the box model in action with an h1 and a div element.

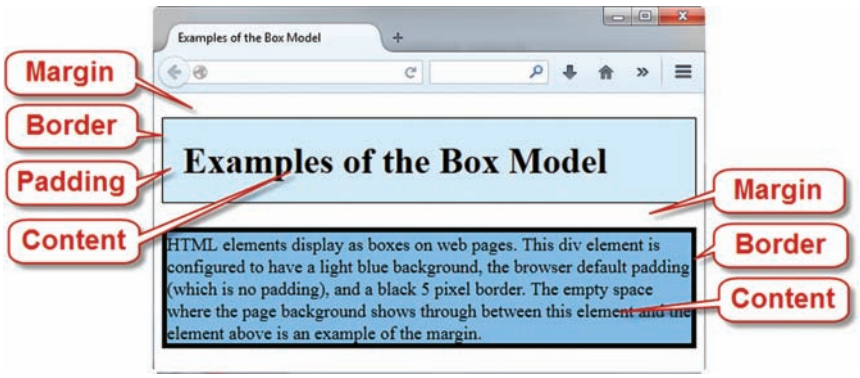

Figure 6.2 Examples of the box model

- The h1 element is configured to have a light-blue background, 20 pixels of padding (the space between the content and the border), and a black, 1 pixel border.
- The empty space where the white web page background shows through is the margin. When two vertical margins meet (such as between the h1 element and the div element), the browser collapses the margin size to be the larger of the two margin values instead of applying both margins.
- The div element has a medium-blue background; the browser default padding (which is no padding); and a black, 5 pixel border.

You will get more practice using the box model in this chapter. Feel free to experiment with the box model and the chapter6/box.html file.

6.2 Normal Flow

Browsers render your web page code line by line in the order it appears in the .html document. This processing is called normal flow. **Normal flow** displays the elements on the page in the order they appear in the web page source code.

Figures 6.3 and 6.4 each display two div elements that contain text content. Let's take a closer look. Figure 6.3 shows a screenshot of two div elements placed one after another on a web page. In Figure 6.4, the boxes are nested inside each other. In both cases, the 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 has 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 CSS positioning and float to configure the flow, or placement, of elements on a web page.

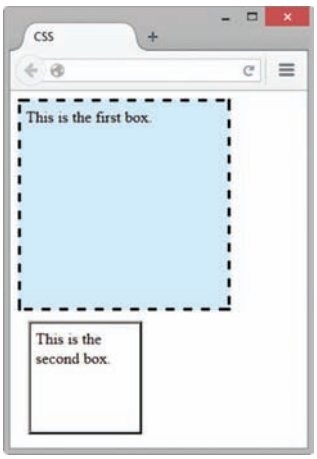

Figure 6.3 Two div elements

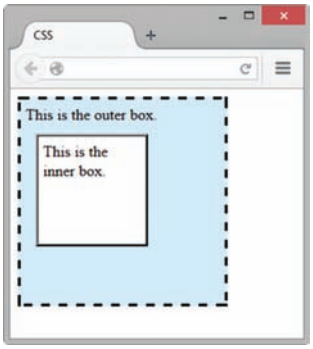

Figure 6.4 Nested div elements

Hands-On Practice 6.1

You will explore the box model and normal flow in this Hands-On Practice as you work with the web pages shown in Figure 6.3 and Figure 6.4.

Practice with Normal Flow

Launch a text editor and open chapter6/starter1.html in the student files. Save the file with the name box1.html. 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 embedded CSS in the head section to configure the "boxes." Add a new style rule for a class named `div1` to configure a light-blue background, dashed border, width of 200 pixels, height of 200 pixels, and 5 pixels of padding. The code is

```
.div1 { width: 200px;
        height: 200px;
        background-color: #D1ECFF;
        border: 3px dashed #000000;
        padding: 5px; }
```

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

```
.div2 { width: 100px;
        height: 100px;
        background-color: #ffffff;
        border: 3px ridge #000000;
        margin: 10px;
        padding: 5px; }
```

Save the file. Launch a browser and test your page. It should look similar to the one shown in Figure 6.3. The student files contain a sample solution (see chapter6/6.1/box1.html).

Practice with Normal Flow and Nested Elements

Launch a text editor and open your box1.html file. Save the file as box2.html.

Edit the code. 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 6.4. Notice how the browser renders the nested div elements: The second box is nested inside the first box because it is coded inside the first div element in the web page source code. This is an example of normal flow. The student files contain a sample solution (see chapter6/6.1/box2.html). The examples in this Hands-On Practice happened to use two div elements. However, the box model applies to block display HTML elements in general, not just to div elements. You will get more practice using the box model in this chapter.

6.3 CSS Float

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 to either the right or left as far as possible within their container (usually either the browser viewport or a div element).

- Use `float: right;` to float the element on the right side of the container.
- Use `float: left;` to float the element on the left side of the container.
- Specify a width for a floated element unless the element already has an implicit width, such as an img element.
- Other elements and web page content will flow around the floated element.

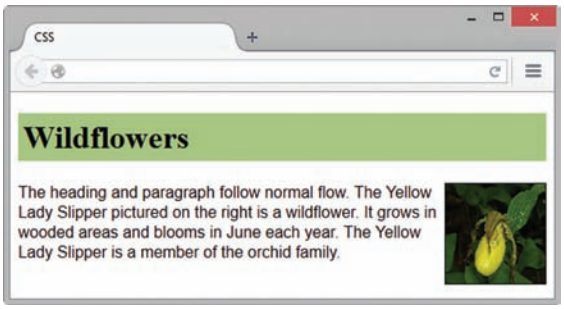

Figure 6.5 The image is configured to float

Figure 6.5 shows a web page with an image configured with `float: right;` to float on the right side of the browser viewport (see chapter6/float.html in the student files). When floating an image, the margin property is useful to configure empty space between the image and text on the page.

View Figure 6.5 and notice how the image stays on the right side of the browser viewport. 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 is

```
h1 { background-color: #A8C682;
     padding: 5px;
     color: #000000; }
p { font-family: Arial, sans-serif; }
#yls { float: right;
       margin: 0 0 5px 5px;
       border: 1px solid #000000; }
```

The HTML source code is

```
<h1>Wildflowers</h1>
<img  id="yls"  src="yls.jpg" alt="Yellow Lady Slipper" height="100"
width="100">
   <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>
```

Hands-On Practice 6.2

In this Hands-On Practice, you'll use the CSS float property as you configure the web page shown in Figure 6.6.

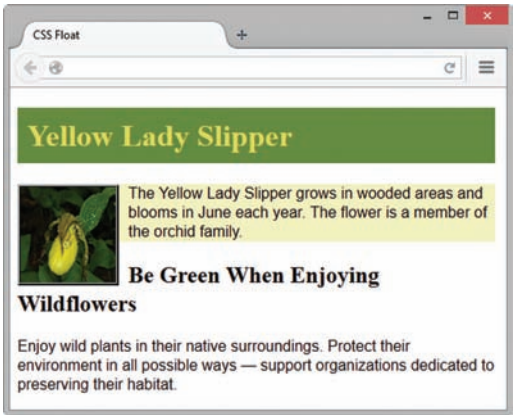

Figure 6.6 The CSS float property left-aligns the image

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

Let's add CSS to float the image. Save the file as index.html in your ch6float folder. Open the file in a text editor and modify the code as follows:

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

```
.float { float: left;
         margin-right: 10px;
         border: 3px ridge #000000; }
```

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 6.6. The student files (chapter6/6.2/index.html) contain a sample solution.

The Floated Element and Normal Flow

Take a moment to examine your file in a browser (see Figure 6.6) and consider how the browser rendered the page. The div element is configured with a light background color to demonstrate how floated elements are rendered outside of normal flow. Observe that the image and the first paragraph are contained within the div element. The h2 element follows the div. If all the elements were rendered using normal flow, the area with the light background color would contain both child elements of the div: the image and the first paragraph. In addition, the h2 element would be placed on its own line under the div element.

However, once the image is placed vertically on the page, it is floated outside of normal flow—that's why the light background color only appears behind the first paragraph and why the h2 element's text begins immediately after the first paragraph and appears next to the floated image. In the following sections, you'll explore properties that can "clear" this float and improve the display.

6.4 CSS: Clearing a Float

The clear Property

The **`clear` property** is often used to terminate, or clear, a float. You can set the value of the clear property to `left`, `right`, or `both`, depending on the type of float you need to clear.

Review Figure 6.6 and the code sample (see chapter6/6.2/index.html in the student files). Notice that although the div element contains both an image and the first paragraph, the light background color of the div only displays behind the screen area occupied by the first paragraph—it stops a bit earlier than expected. Clearing the float will help take care of this display issue.

Clear a Float with a Line Break

A common technique to clear a float within a container element is to add a line break element configured with the `clear` property. See chapter6/clear1.html in the student files for an example. Observe that a CSS class is configured to clear the left float:

```
.clearleft { clear: left; }
```

Also, a line break tag assigned to the `clearleft` class is coded before the closing `div` tag. The code for the div element is

```
<div>
<img class="float" src="yls.jpg" alt="Yellow Lady Slipper"
height="100" width="100">
<p>The Yellow Lady Slipper grows in wooded areas and blooms in June
each year. The flower is a member of the orchid family.</p>
<br  class="clearleft">
</div>
```

Figure 6.7 displays a screen shot of this page. Review Figure 6.6 and note two changes: the light background color of the div element extends farther down the page and the h2 element's text begins on its own line under the image.

Another Technique to Clear a Float

If you are not concerned about the light background color display, another option is to omit the line break tag and, instead, apply the `clearleft` class to the h2 element, which is the first block display element after the div. This does not change the display of the light background color, but it does force the h2 element's text to begin on its own line, as shown in Figure 6.8 (see chapter6/clear2.html in the student files).

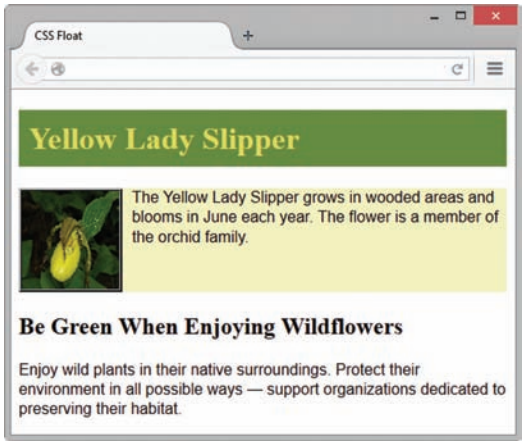

Figure 6.7 The clear property is applied to a line break tag

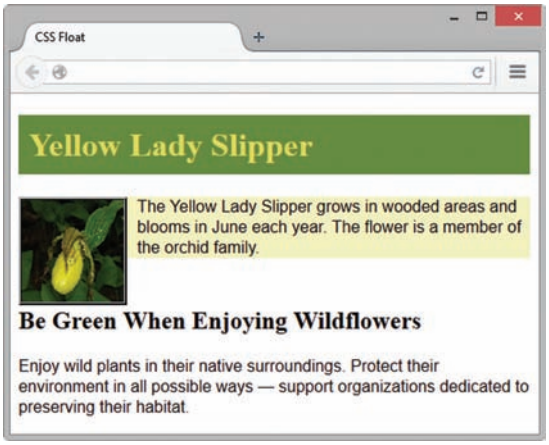

Figure 6.8 The clear property is applied to the h2 element

The overflow Property

The **`overflow` property** is often used to clear a float, although its intended purpose is to configure how content should display if it is too large for the area allocated. See Table 6.2 for a list of commonly used values for the overflow property.

Table 6.2 The overflow property

Value	Purpose
`visible`	Default value; the content is displayed, and if it's too large, the content will overflow and extend outside the area allocated to it
`hidden`	The content is clipped to fit the area allocated to the element in the browser viewport
`auto`	The content fills the area allocated to it and, if needed, scrollbars are displayed to allow access to the remaining content
`scroll`	The content is rendered in the area allocated to it and scrollbars are displayed

Clear a Float

Review Figure 6.6 and the code sample (chapter6/6.2/index.html in the student files). Observe the div element, which contains the floated image and first paragraph on the page. Notice that although the div element contains both an image and the first paragraph, the light background color of the div element does not extend as far as expected; it is only visible in the area occupied by the first paragraph. You can use the `overflow` property assigned to the container element to resolve this display issue and clear the float. In this case, we'll apply the `overflow` and `width` properties to the div selector. The CSS to configure the div in this manner is

```
div { background-color: #F3F1BF;
      overflow: auto;
      width: 100%; }
```

This will clear the float. The web page will display as shown in Figure 6.9 (see chapter6/overflow.html in the student files).

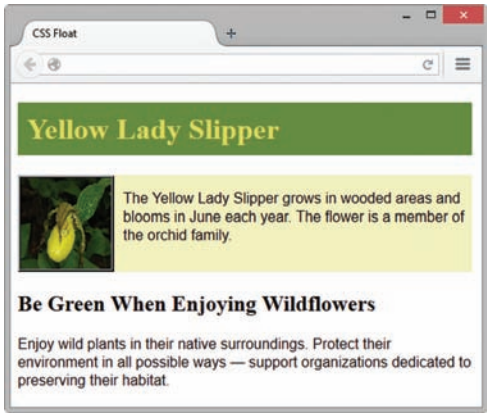

Figure 6.9 The overflow property is applied to the div selector

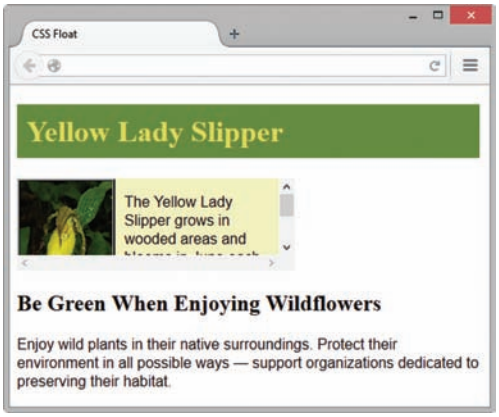

Figure 6.10 The browser displays scrollbars

Notice that using the `overflow` property (see Figure 6.9) and applying the `clear` property to a line break tag (see Figure 6.7) result in a similar web page display. You may be wondering about which CSS property (`clear` or `overflow`) is the best one to use when you need to clear a float.

Although the `clear` property is widely used, in this example, it is more efficient to apply the `overflow` property to the container element (for example, a div element). This will clear the float, avoid adding an extra line break tag, and ensure that the container element expands to enclose the entire floated element. You'll get more practice with the `float`, `clear`, and `overflow` properties as you continue working through this textbook. Floating elements is a key technique for designing multicolumn page layouts with CSS.

Configure Scrollbars

The web page in Figure 6.10 demonstrates the use of `overflow: auto;` to automatically display scrollbars if the content exceeds the space allocated to it. In this case, the div that contains the paragraph and the floated image was configured with a width of 300px and a height of 100px.

See the example web page (chapter6/scroll.html in the student files). The CSS for the div is shown below:

```
div { background-color: #F3F1BF;
      overflow: scroll;
      width: 300px;
      height: 100px; }
```

Checkpoint 6.1

1. List the components of the box model from innermost to outermost.
2. Describe the purpose of the CSS float property.
3. Which two CSS properties can be used to clear a float?

6.5 CSS Box Sizing

When you view an element on a web page it's intuitive to expect that the width of an element on a page includes the size of the element's padding and border. However, this isn't the default behavior of browsers. Recall from the box model introduction in Section 6.1 that the `width` property by default only includes the actual width of the content itself within the element and does not also include the width of any padding or border that may exist for the element. The purpose of the `box-sizing` property is to alleviate this issue. The **`box-sizing` property** causes the browser calculation of the width or height to include the content's actual width or height in addition to the width or height of any padding and border that may exist.

Valid `box-sizing` property values include `content-box` (the default) and `border-box`. Use the CSS `box-sizing: border-box;` declaration to configure the browser to also include the values of the border and padding when calculating the width and height properties of an element.

Figures 6.11 and 6.12 show web pages (chapter6/boxsizing1.html and chapter6/boxsizing2.html in the student files) that each have floated elements configured with 30% width, 150px height, 20px padding, and 10px margin. The page in Figure 6.11 uses default box-sizing. The page in Figure 6.12 uses box-sizing set to `border-box`. The size of the elements and the placement of the elements on the pages differ. You may notice at first glance that the elements look larger in Figure 6.11. The larger display is because the browser sets the content to 30% width before adding the 20 pixels of padding on each side. The elements are smaller in Figure 6.12. The smaller display is because the browser applies the 30% width to the combination of the padding and the content.

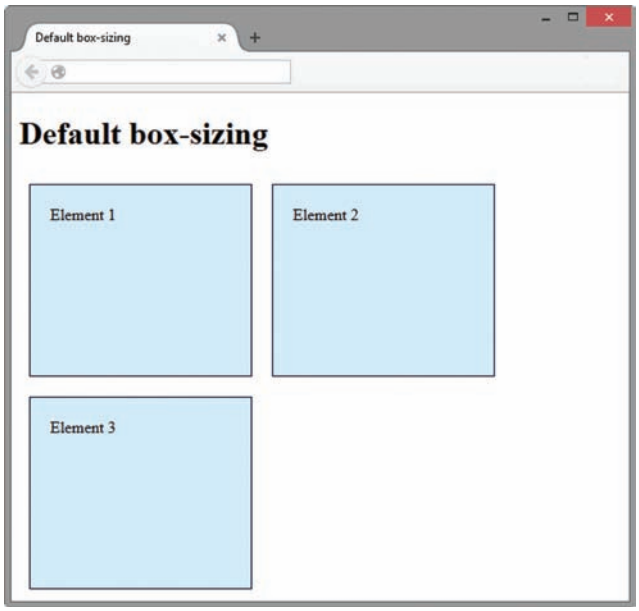

Figure 6.11 Default box-sizing

Let's take a closer look at the placement of the three floated elements on the pages. Figure 6.11 does not display all three elements side-by-side. This web page uses default `box-sizing` so the browser assigned the 30% width to each element's content only and then added 20 pixels of padding to each side of each element. Due to these calculations, the browser determined there was not enough room in the browser viewport to display all three elements next to each other and the browser dropped the third floated element to the next line. The web page in Figure 6.12 is coded with `box-sizing` set to `border-box` which configures the three floated elements to be displayed side-by-side because the browser assigned the 30% width to the combined content and padding areas (including 20 pixels of padding on each side).

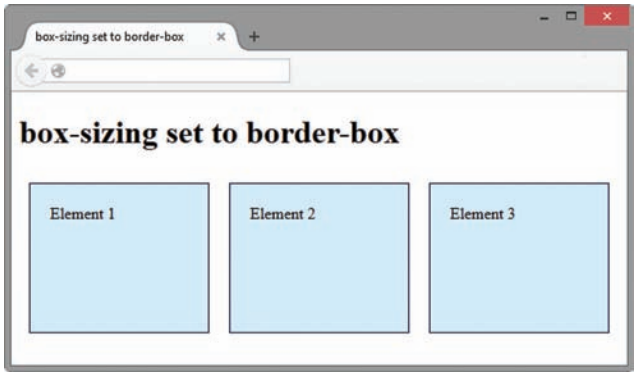

Figure 6.12 box-sizing set to border-box

It is common practice for web developers to apply border-box box-sizing when they plan to use floated elements or multicolumn layouts. It's also common practice to apply `box-sizing` by configuring

the * **universal selector**, which will target all HTML elements. The CSS style rule to apply border-box box-sizing to all elements with the universal selector is

```
* { box-sizing: border-box; }
```

Feel free to experiment with the box-sizing property and the example files (chapter6/boxsizing1.html and chapter6/boxsizing2.html in the student files).

6.6 CSS Two-Column Layout

Your First Two-Column Layout

A common design for a web page is a two-column layout. This is accomplished with CSS by configuring one of the columns to float on the web page. Coding HTML is a skill and skills are best learned by practice. The next Hands-On Practice guides you as you convert a single-column page layout (Figure 6.13) into your first two-column layout (Figure 6.14).

Figure 6.13 Single-column layout.

Figure 6.14 Two-column layout.

Hands-On Practice 6.3

A. **Review single-column layout.** Launch a text editor, and open the singlecol.html file from the chapter6 folder in the student files. Take a moment to look over the code. Notice the structure of the HTML tags correspond to wireframe in Figure 6.13.

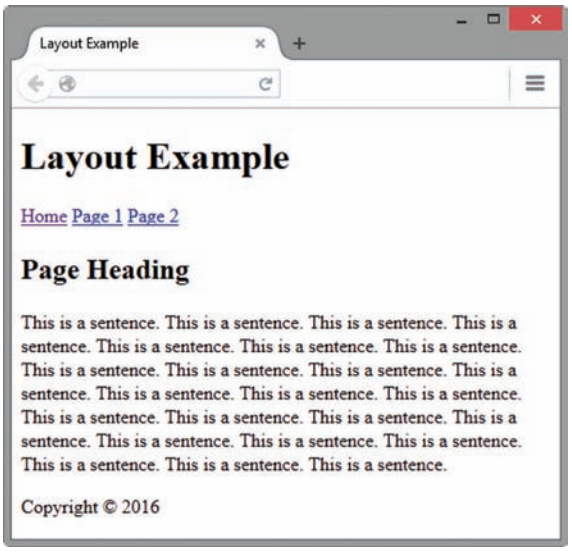

Figure 6.15 Web page with single-column layout.

```
<body>
<div id="wrapper">
  <header> </header>
  <nav> </nav>
  <main> </main>
  <footer> </footer>
</div>
</body>
```

Save the file as index.html. When you display index.html in a browser, your display should be similar to Figure 6.15.

B. **Configure a two-column layout.** Launch a text editor and open the index.html file. You will edit the HTML and CSS to configure a two-column layout as shown in Figure 6.14 wireframe.

1. **Edit the HTML.** The single-column navigation is horizontal but the two-column navigation will be displayed in a vertical orientation. Later in this chapter, you'll learn how to configure navigation hyperlinks within an unordered list but for now, a quick adjustment is to code a line break tag after each of the first two hyperlinks in the nav area.

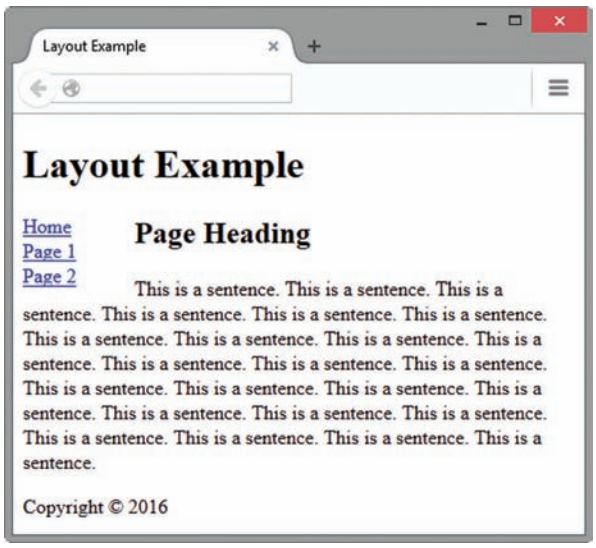

Figure 6.16 The nav is floating on the left.

2. **Configure the float with CSS.** Locate the style tags in the head section of the document and code the following style rule as embedded CSS to configure a nav element with a width of 90px that floats to the left.

```
nav { float: left;
      width: 90px; }
```

Save the file and test it in the Firefox or Chrome browser. Your display will be similar to Figure 6.16. Notice that the content in the main area wraps around the floated nav element.

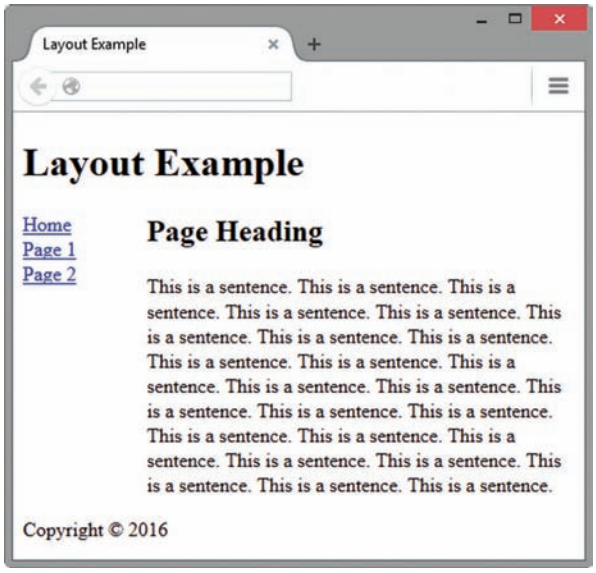

Figure 6.17 Two-column layout.

3. **Configure two columns with CSS.** You just configured the nav element to float on the left. The main element will be in the right-side column and will be configured with a left margin (the same side as the float). To get a two-column look, the value of the margin should be greater than the width of the floated element. Open the index.html file in a text editor and code the following style rule to configure a 100px left margin for the main element.

```
main { margin-left: 100px; }
```

Save the file and test it in the Firefox or Chrome browser. Your display will be similar to Figure 6.17 with a two-column layout.

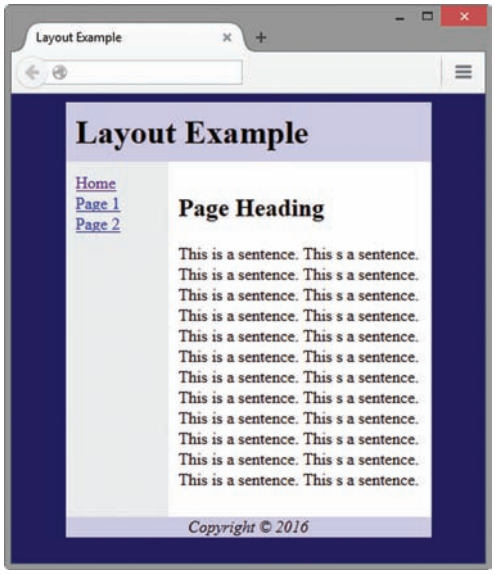

Figure 6.18 Final two-column layout.

4. **Enhance the page with CSS.** Code the following style rules as embedded CSS to create a more appealing web page. When you have completed this step, your page should be similar to Figure 6.18.

   a. **The body element selector.** Configure a dark background color.

   ```
   body { background-color:
   #000066; }
   ```

   b. **The wrapper id selector.** Configure 80% width, centered on the page, and a light (#EAEAEA) background color. This background color will display behind child elements (such as the nav element) that do not have a background color configured.

   ```
   #wrapper { width: 80%;
              margin-left: auto;
              margin-right: auto;
              background-color: #EAEAEA; }
   ```

   c. **The header element selector.** Configure #CCCCFF background color.

   ```
   header { background-color: #CCCCFF; }
   ```

   d. **The h1 element selector.** Configure 0 margin and 10px of padding.

   ```
   h1 { margin: 0;
        padding: 10px; }
   ```

   e. **The nav element selector.** Edit the style rule and add a declaration for 10 pixels of padding.

   ```
   nav { float: left;
         width: 90px;
         padding: 10px;  }
   ```

   f. **The main element selector.** Edit the style rule and add a declaration for 10 pixels of padding and #FFFFFF background color.

   ```
   main { margin-left: 100px;
          padding: 10px;
          background-color: #FFFFFF;  }
   ```

   g. **The footer element selector.** Configure centered, italic text, and a #CCCCFF background color. Also configure the footer to clear all floats.

   ```
   footer { text-align: center;
            font-style: italic;
            background-color: #CCCCFF;
            clear: both; }
   ```

Save your file and test it in the Firefox or Chrome browser. Your display should be similar to Figure 6.18. You can compare your work to the sample in the student files (chapter6/6.3/index.html). At the time this was written, Internet Explorer did not support default styles like the HTML5 main element. You may need to nudge this browser to comply by adding the `display: block;` declaration (introduced later in this chapter) to the styles for the main element selector. An example solution is in the student files (chapter6/6.3/iefix.html).

Two-Column Layout Example

The web page you coded in Hands-On Practice 6.3 is just one example of a two-column layout design. Let's explore coding the two-column layout with a footer in the right column as shown in Figure 6.19 wireframe. The HTML template for the page layout is

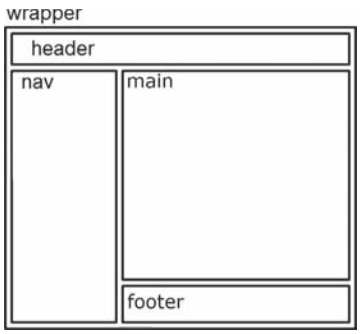

Figure 6.19 Alternate wireframe.

```
<div id="wrapper">
  <header>
  </header>
  <nav>
  </nav>
  <main>
  </main>
  <footer>
  </footer>
</div>
```

The key CSS configures a floating nav element, a main element with a left margin, and a footer with a left margin.

```
nav { float: left; width: 150px; }
main { margin-left: 165px; }
footer { margin-left: 165px; }
```

The web page shown in Figure 6.20 implements this layout. An example is in the student files, chapter6/layout/twocol.html.

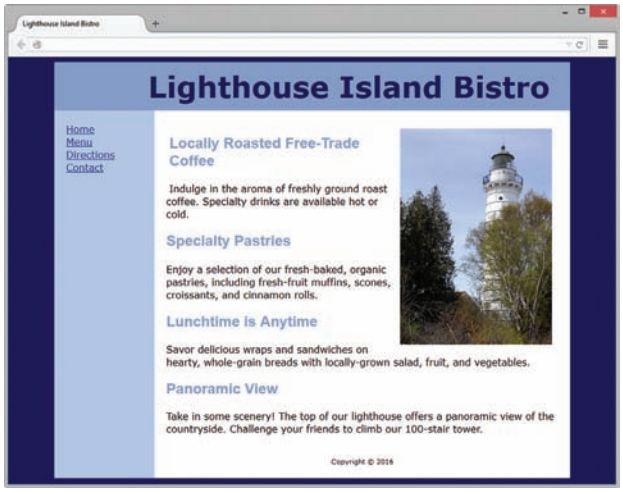

Figure 6.20 Page with alternate layout.

FAQ Does the order of the floated and non-floated elements matter?

A key to coding successful layouts with float is in the HTML—code the element that needs to float BEFORE its companion elements. The browser will shift the floated element over to the side of the browser viewport and display the elements that follow alongside the floated element.

FAQ Do I have to use a wrapper?

No, you are not required to use a wrapper or container for a web page layout. However, it does make it easier to get the two-column look because the background color of the wrapper div will display behind any of its child elements that do not have their own background color configured. This technique also provides you with the option of configuring a different background color or background image for the page using the body element selector.

Not Yet Ready for Prime Time

There is one more aspect of the two-column layout web page design to consider before it is ready for "prime time." The navigation area is a list of hyperlinks. In order to more closely semantically describe the navigation area, the hyperlinks should be configured in an unordered list. In the next section, you'll learn techniques to configure horizontal and vertical navigation hyperlinks in unordered lists.

6.7 Hyperlinks in an Unordered List

One of the advantages of using CSS for page layout involves the use of semantically correct code. Writing semantically correct code means choosing the markup tag that most accurately reflects the purpose of the content. Using the various levels of heading tags for content headings and subheadings or placing paragraphs of text within paragraph tags (rather than using line breaks) are examples of writing semantically correct code. This type of coding is a step in the direction of supporting the Semantic Web. Leading Web developers such as Eric Meyer, Mark Newhouse, Jeffrey Zeldman, and others have promoted the idea of using unordered lists to configure navigation menus. After all, a navigation menu is a list of hyperlinks.

Focus on Accessibility

Configuring navigation with a list also helps to provide accessibility. Screen reader applications offer easy keyboard access and verbal cues for information organized in lists, such as the number of items in the list.

Configure List Markers with CSS

Recall that the default display for an unordered list is to show a disc marker (often referred to as a bullet) in front of each list item. The default display for an ordered list is to show a decimal number in front of each list item. When you configure navigation hyperlinks in an unordered list, you may not always want to see those **list markers**. It's easy to configure them with CSS. Use the **`list-style-type` property** to configure the marker for an unordered or ordered list. See Table 6.3 for common property values.

Table 6.3 CSS properties for ordered and unordered list markers

Property	Description	Value	List Marker Display
`list-style-type`	Configures the style of the list marker	`none`	No list markers display
		`disc`	Circle
		`circle`	Open circle
		`square`	Square
		`decimal`	Decimal numbers
		`upper-alpha`	Uppercase letters
		`lower-alpha`	Lowercase letters
		`lower-roman`	Lowercase Roman numerals
`list-style-image`	Image replacement for the list marker	The `url` keyword with parentheses surrounding the file name or path for the image	Image displays in front of each list item
`list-style-position`	Configures placement of the list marker	`inside`	Markers are indented; text wraps under the markers
		`outside` (default)	Markers have default placement

Figure 6.21 shows an unordered list configured with square markers using the following CSS:

```
ul { list-style-type: square; }
```

Figure 6.22 shows an ordered list configured with uppercase letter markers using the following CSS:

```
ol { list-style-type: upper-alpha; }
```

Figure 6.21 The unordered list markers are square

Figure 6.22 The ordered list markers use uppercase letters

Figure 6.23 The list markers are replaced with an image

Configure an Image as a List Marker

Use the **`list-style-image` property** to configure an image as the marker in an unordered or ordered list. In Figure 6.23, an image named trillium.gif was configured to replace the list markers using the following CSS:

```
ul {list-style-image: url(trillium.gif); }
```

Vertical Navigation with an Unordered List

Figure 6.24 shows the navigation area of a web page (see chapter6/layout/twocolnav.html in the student files) that uses an unordered list to organize the navigation links. The HTML is

```
<ul>
<li><a href="index.html">Home</a></li>
<li><a href="menu.html">Menu</a></li>
<li><a href="directions.html">Directions</a></li>
<li><a href="contact.html">Contact</a></li>
</ul>
```

Figure 6.24 Navigation in an unordered list

Configure with CSS

OK, so now that we're semantically correct, how about improving the visual aesthetic? Let's use CSS to eliminate the list marker. We also need to make sure that our special styles only apply to the unordered list in the navigation area (within the `nav` element), so we'll use a descendant selector. The CSS to configure the list in Figure 6.25 is

```
nav ul { list-style-type: none; }
```

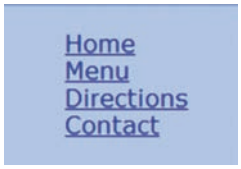

Figure 6.25 The list markers have been eliminated with CSS

Remove the Underline with CSS

The **`text-decoration` property** modifies the display of text in the browser. This property is most often used to eliminate the underline from the navigation hyperlinks with `text-decoration: none;`.

The CSS to configure the list in Figure 6.26 (see chapter6/layout/twocolnav2.html in the student files) that eliminates the underline on the hyperlinks in the navigation area (within the `nav` element) is

```
nav a { text-decoration: none; }
```

Figure 6.26 The CSS text-decoration property has been applied

Horizontal Navigation with an Unordered List

You may be wondering how to use an unordered list for a horizontal navigation menu. The answer is CSS! List items are block display elements. They need to be configured as inline display elements to display in a single line. The CSS **`display` property** configures how the browser renders, or displays, an element on a web page. See Table 6.4 for a list of commonly used values.

Table 6.4 The display property

Value	Purpose
`none`	The element will not display
`inline`	The element will display as an inline element in the same line as the surrounding text and/or elements
`inline-block`	The element will display as an inline display element adjacent to other inline display elements but also can be configured with properties of block display elements including width and height.
`block`	The element will display as a block element with a margin above and below

Figure 6.27 Navigation in an unordered lisxt

Figure 6.27 shows the navigation area of a web page (see chapter6/layout/horizontal.html in the student files) with a horizontal navigation area organized in an unordered list. The HTML is

```
<nav>
<ul>
   <li><a href="index.html">Home</a></li>
   <li><a href="menu.html">Menu</a></li>
   <li><a href="directions.html">Directions</a></li>
   <li><a href="contact.html">Contact</a></li>
</ul>
</nav>
```

Configure with CSS

The following CSS was applied in this example:

- To eliminate the list marker, apply `list-style-type: none;` to the ul element selector:

```
nav ul { list-style-type: none; }
```

- To render the list items horizontally instead of vertically, apply `display: inline;` to the li element selector:

```
nav li { display: inline; }
```

- To eliminate the underline from the hyperlinks, apply `text-decoration: none;` to the a element selector. Also, to add some space between the hyperlinks, apply `padding-right: 10px;` to the a element selector:

```
nav a { text-decoration: none; padding-right: 10px; }
```

6.8 CSS Interactivity with Pseudo-Classes

Have you ever visited a website and found that the text hyperlinks changed color when you moved the mouse pointer over them? Often, this is accomplished using a CSS **pseudo-class**, which can be used to apply a special effect to a selector. The five pseudo-classes that can be applied to the anchor element are shown in Table 6.5.

VideoNote
Interactivity with CSS pseudo-classes

Table 6.5 Commonly used CSS pseudo-classes

Pseudo-Class	When Applied
`:link`	Default state for a hyperlink that has not been clicked (visited)
`:visited`	Default state for a visited hyperlink
`:focus`	Triggered when the hyperlink has keyboard focus
`:hover`	Triggered when the mouse pointer moves over the hyperlink
`:active`	Triggered when the hyperlink is actually clicked

Notice the order in which the pseudo-classes are listed in Table 6.5. Anchor element pseudo-classes *must be coded in this order* (although it's OK to omit one or more of those listed). If you code the pseudo-classes in a different order, the styles will not be reliably applied. It's common practice to configure the `:hover`, `:focus`, and `:active` pseudo-classes with the same styles.

To apply a pseudo-class, write it after the selector. The following code sample will configure text hyperlinks to be red initially. The sample also uses the `:hover` pseudo-class to configure the hyperlinks to change their appearance when the visitor places the mouse pointer over them so that the underline disappears and the color changes.

```
a:link { color: #ff0000; }
a:hover { text-decoration: none;
          color: #000066; }
```

Figure 6.28 shows part of a web page that uses a similar technique. Note the position of the mouse pointer over the Print This Page hyperlink. The text color has changed and has no underline.

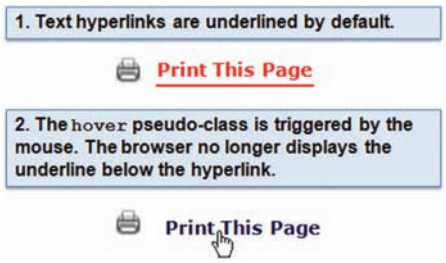

Figure 6.28 Using the `hover` pseudo-class

Hands-On Practice 6.4

You will use pseudo-classes to create interactive hyperlinks in this Hands-On Practice. Create a folder named ch6hover. Copy the lighthouseisland.jpg, lighthouselogo.jpg, and starter.html files from the chapter6 folder in the student files into your ch6hover folder. Display the web page in a browser. It should look similar to Figure 6.29. Notice that the navigation area needs to be configured.

Launch a text editor and open starter.html. Save the file as index.html in your ch6hover folder.

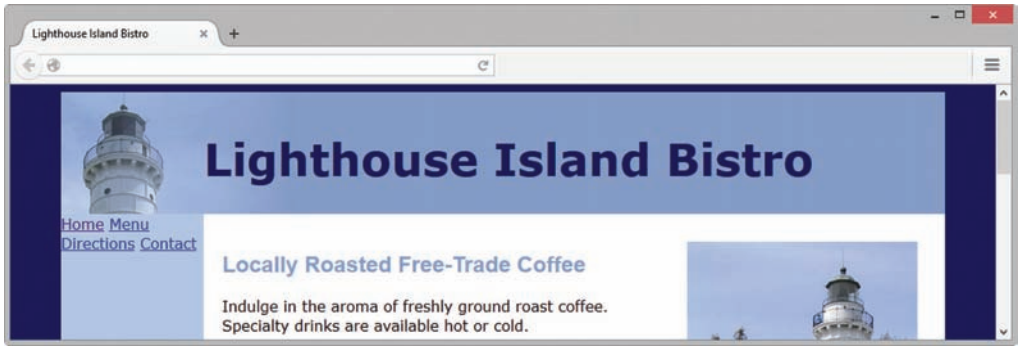

Figure 6.29 The navigation area needs to be styled in this two-column page layout

1. Review the code for this page, which uses a two-column layout. Examine the nav element and modify the code to configure the navigation in an unordered list.

```
<ul>
   <li><a href="index.html">Home</a></li>
   <li><a href="menu.html">Menu</a></li>
   <li><a href="directions.html">Directions</a></li>
   <li><a href="contact.html">Contact</a></li>
</ul>
```

   Let's add CSS to the embedded styles to configure the unordered list elements in the nav element. Eliminate the list marker, set the left margin to 0, and set the padding to 10 pixels.

```
nav ul { list-style-type: none;
         margin-left: 0;
         padding: 10px; }
```

2. Next, configure basic interactivity with pseudo-classes.
   - Configure the anchor tags in the nav element to have 10 pixels of padding, use bold font, and display no underline.

```
nav a { text-decoration: none;
        padding: 10px;
        font-weight: bold; }
```

   - Use pseudo-classes to configure anchor tags in the nav element to display white (#ffffff) text for unvisited hyperlinks, light-gray (#eaeaea) text for visited hyperlinks, and dark-blue (#000066) text when the mouse pointer hovers over hyperlinks:

```
nav a:link { color: #ffffff; }
nav a:visited { color: #eaeaea; }
nav a:hover { color: #000066; }
```

Save your page and test in a browser. Move your mouse pointer over the navigation area and notice the change in the text color. Your page should look similar to Figure 6.30 (see chapter6/6.4/index.html in the student files).

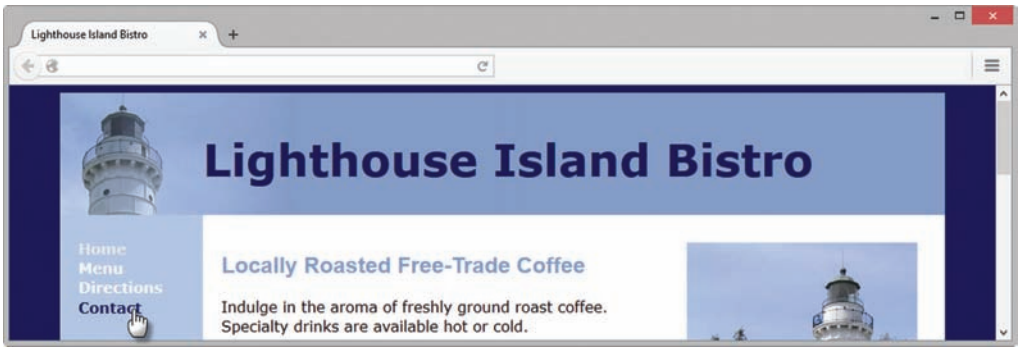

Figure 6.30 CSS pseudo-classes add interactivity to the navigation

CSS Buttons

In addition to configuring an interactive text hyperlink, another use of CSS and pseudo-classes is to configure a hyperlink that looks like a button. This coding technique saves on the bandwidth used by button image files. Figure 6.31 depicts a web page with a "button" configured with CSS instead of with an image.

Figure 6.31 CSS button

Hands-On Practice 6.5

In this Hands-On Practice you will explore the CSS button coding technique and create the web page shown in Figure 6.31. Create a new folder named button. Launch a text editor and open the template file located at chapter2/template.html in the student files. Save the file as index.html in your button folder. Modify the file to configure a web page as indicated:

1. Configure the text, CSS Button, within an h1 element and within the title element.
2. Configure a hyperlink with an anchor tag. Assign the anchor tag to a class named `button`. Configure the hyperlink to display the text, Sign Up, and link to a file named signup.html.
   ```
   <a class="button" href="signup.html">Sign Up</a>
   ```
3. Now, let's add embedded CSS. Code a style element in the head section. Configure the following CSS within the style element.
   a. Configure the body element selector with `text-align: center;`
   b. Configure the h1 element selector with 2em bottom margin.
   ```
   h1 { margin-bottom: 2em; }
   ```

c. Configure the `button` class selector with 60px border radius, 1em padding, #FFFFFF text color, #E38690 background color, Arial or sans-serif font family, 2em size bold font, centered text, and no text decoration. The CSS follows

```
.button { border-radius: 60px;
          padding: 1em;
          color: #FFFFFF;
          background-color: #E38690;
          font-family: Arial, sans-serif;
          font-size: 2em;
          font-weight: bold;
          text-align: center;
          text-decoration: none; }
```

d. Configure the `:link`, `:visited`, and `:hover` pseudo-classes for the `button` class. The CSS follows

```
.button:link    { color : #FFFFFF; }
.button:visited { color : #CCCCCC; }
.button:hover   { color : #965251;
                  background-color: #F7DEE1; }
```

Save your page and display it in a browser. When you place the cursor over the button you should see the text and background color change. A sample solution is in the student files (chapter6/6.5/button.html).

Figure 6.32 (also in the student files chapter6/6.5/navbuttons.html) demonstrates using the CSS button coding technique to configure a group of horizontal navigation buttons.

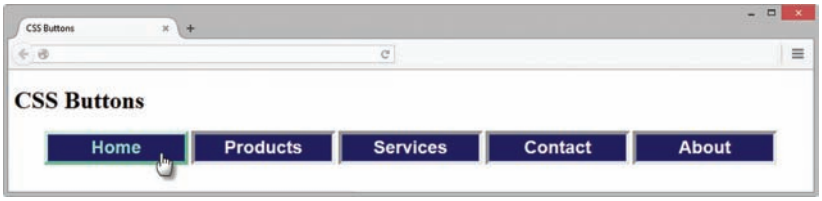

Figure 6.32 Navigation "buttons" configured with CSS

6.9 Practice with CSS Two-Column Layout

You've had some experience with a two-column layout, organizing navigation links in an unordered list, and configuring CSS pseudo-classes. Let's reinforce these skills with a Hands-On Practice.

Hands-On Practice 6.6

In this Hands-On Practice, you'll create a new version of the Lighthouse Island Bistro home page with a header spanning two columns, content in the left column, vertical navigation in the right column, and a footer below the two columns. See Figure 6.33 for a wireframe. You will configure the CSS in an external style sheet. Create a folder named ch6practice. Copy the starter2.html, lighthouseisland.jpg, and lighthouselogo.jpg files from the chapter6 folder in the student files into your ch6practice folder.

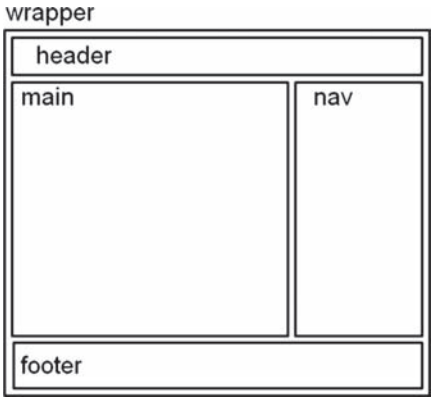

Figure 6.33 Wireframe for a two-column layout with navigation in the right-side column.

1. Launch a text editor and open starter2.html. Save the file as index.html. Add a link element to the head section of the web page that associates this file with an external style sheet named lighthouse.css. A code sample follows.

```
<link href="lighthouse.css" rel="stylesheet">
```

2. Save the index.html file. Launch a text editor and create a new file named lighthouse.css in your ch6practice folder. Configure the CSS for the wireframe sections as follows:

   - The universal selector: set the box-sizing property to border-box.

```
* { box-sizing: border-box; }
```

   - The body element selector: very-dark-blue background (#00005D) and Verdana, Arial, or the default sans-serif font typeface

```
body { background-color: #00005D;
       font-family: Verdana, Arial, sans-serif; }
```

   - The `wrapper` id: centered, width set to 80% of the browser viewport, minimum width of 960px, maximum width of 1200px, display text in a dark-blue color (#000066), and display a medium-blue (#B3C7E6) background color (*this color will display behind the nav area*)

```
#wrapper { margin: 0 auto;
           width: 80%;
           min-width: 960px; max-width: 1200px;
           background-color: #B3C7E6;
           color: #000066; }
```

   - The header element selector: slate-blue (#869DC7) background color; very-dark-blue (#00005D) text color; 150% font size; top, right, and bottom padding of 10px; 155 pixels of left padding; and the lighthouselogo.jpg background image

```
header { background-color: #869DC7;
         color: #00005D;
         font-size: 150%;
         padding: 10px 10px 10px 155px;
         background-repeat: no-repeat;
         background-image: url(lighthouselogo.jpg); }
```

- The nav element selector: float on the right, width set to 200px, display bold text, letter spacing of 0.1 em

```
nav { float: right;
      width: 200px;
      font-weight: bold;
      letter-spacing: 0.1em; }
```

- The main element selector: white background color (#FFFFFF), black text color (#000000), 10 pixels of padding on the top and bottom, 20 pixels of padding on the left and right, display set to block (to prevent an Internet Explorer display issue), and overflow set to auto

```
main { background-color: #ffffff;
       color: #000000;
       padding: 10px 20px; display: block;
       overflow: auto; }
```

- The footer element selector: 70% font size, centered text, 10 pixels of padding, slate-blue background color (#869DC7), and clear set to both

```
footer { font-size: 70%;
         text-align: center;
         padding: 10px;
         background-color: #869DC7;
         clear: both;}
```

Save the file and display it in a browser. Your display should be similar to Figure 6.34.

Figure 6.34 Home page with major sections configured using CSS

3. Continue editing the lighthouse.css file to style the h2 element selector and floating image. Configure the h2 element selector with slate-blue text color (#869DC7) and Arial or sans-serif font typeface. Configure the `floatright` id to float on the right side with 10 pixels of margin.

```
h2 { color: #869DC7;
     font-family: Arial, sans-serif; }
#floatright { float: right;
              margin: 10px; }
```

4. Continue editing the lighthouse.css file and configure the navigation bar.

- The ul element selector: Eliminate the list markers. Set a zero margin and zero padding.

```
nav ul { list-style-type: none; margin: 0; padding: 0; }
```

- The a element selector: no underline, 20 pixels of padding, medium-blue background color (#B3C7E6), and 1 pixel solid white bottom border

Use `display: block;` to allow the web page visitor to click anywhere on the anchor "button" to activate the hyperlink.

```
nav a { text-decoration: none;
        padding: 20px;
        display: block;
        background-color: #B3C7E6;
        border-bottom: 1px solid #FFFFFF; }
```

Configure the `:link`, `:visited`, and `:hover` pseudo-classes as follows:

```
nav a:link { color: #FFFFFF; }
nav a:visited { color: #EAEAEA; }
nav a:hover { color: #869DC7;
              background-color: #EAEAEA; }
```

Save your files. Open your index.html file in a browser. Move your mouse pointer over the navigation area and notice the interactivity, as shown in Figure 6.35 (see chapter6/6.6/index.html in the student files).

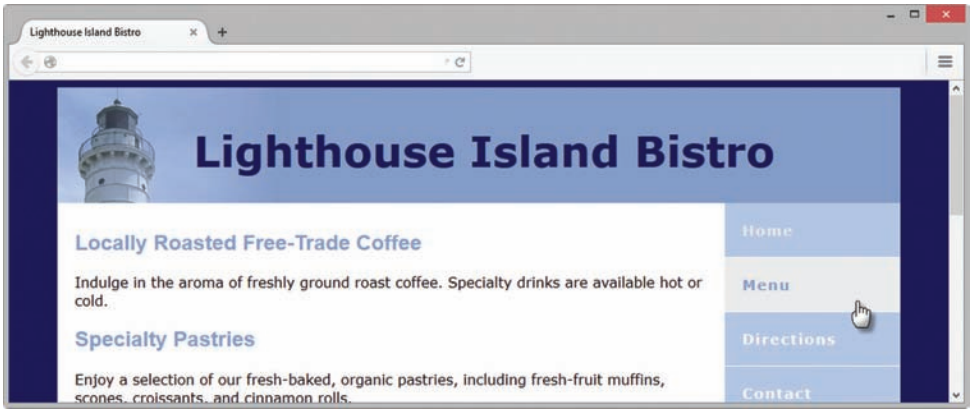

Figure 6.35 CSS pseudo-classes add interactivity to the page

6.10 Header Text Image Replacement

Sometimes a client may have a specific font typeface other than a web-safe font that they would like to use in the header section for the company name. In this situation, it's common for a graphic designer to configure a header banner image that displays the company name in a font that is not web-safe, as shown in Figure 6.36.

Figure 6.36 Header banner image with text in Papyrus, a non web-safe font

It is acceptable to display the banner with an image tag or configure the banner as a background image within the header element. However, the text contained in the image is not available to assistive technology such as screen readers and is not immediately visible to search engines.

A popular technique called **Header Text Image Replacement**, described by web designer Chris Coyier at http://css-tricks.com/header-text-image-replacement, allows you to configure a header banner image that displays text *and* an h1 element that contains text. The key is that the h1 text does not display within the browser viewport but is available to assistive technology and search engines. It's the best of both worlds, the client's banner image is displayed and plain text is available to provide for accessibility and search engine optimization.

To apply Header Text Image Replacement:

1. Code the company or website name with the h1 element.
2. Configure styles for the header element; set the header banner image as the background of the header or h1 element.
3. Use the `text-indent` property to configure placement of the h1 text outside of the browser viewport, the declaration `text-indent: -9999px;` is most commonly used.

Improved Header Text Image Replacement Technique

The Header Text Image Replacement technique works great on desktop and laptop computers. However, there can be a performance issue when web pages using this technique are displayed on mobile devices with slower processors as the browsers actually try to draw a box offscreen that is over 9999 pixels wide. Noted web developer Jeffrey Zeldman shared a method at http://www.zeldman.com/2012/03/01/replacing-the-9999px-hack-new-image-replacement/ that prevents the performance issues. Set the h1 element selector's `text-indent` to 100% instead of to -9999px. Also, configure wrapping and overflow by setting the `white-space` property to `nowrap` and the `overflow` property to `hidden`. You'll practice using this adapted Header Text Image Replacement technique in this Hands-On Practice.

Hands-On Practice 6.7

Create a folder named replacech5. Copy the starter3.html file from the chapter6 folder into the replace folder and rename it as index.html. Copy the background.jpg and lighthousebanner.jpg files from chapter6/starters into your replacech6 folder.

Launch a text editor and open index.html.

1. Examine the HTML. Notice that an h1 element with the text "Lighthhouse Island Bistro" is present.
2. Edit the embedded styles and configure a style rule for the header element selector with height set to 150 pixels, bottom margin set to 0, and lighthousebanner.jpg as a background image that does not repeat.

```
header { height: 150px;
         margin-bottom: 0;
         background-image: url(lighthousebanner.jpg);
         background-repeat: no-repeat; }
```

Save the file and test in a browser. You'll see that both the h1 text and the text on the lighthousebanner.jpg image display, as shown in Figure 6.37.

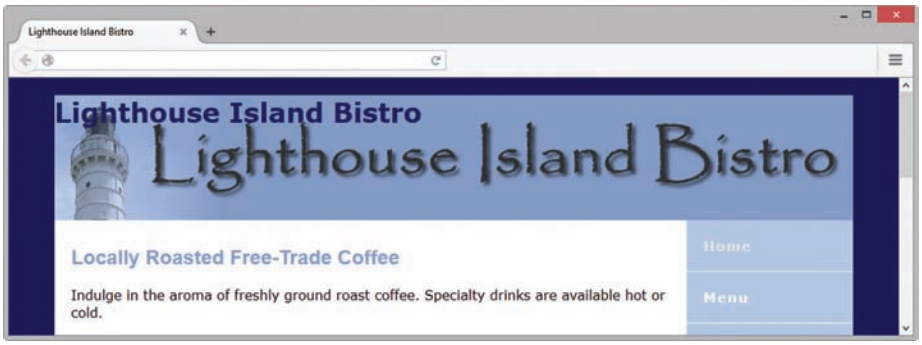

Figure 6.37 Both the h1 text and the text in the image temporarily display

3. Edit the embedded styles and configure the `text-indent`, `white-space`, and `overflow` properties for the h1 element selector as follows:

```
h1 { text-indent: 100%;
     white-space: nowrap;
     overflow: hidden; }
```

Launch a browser and view index.html as shown in Figure 6.38.

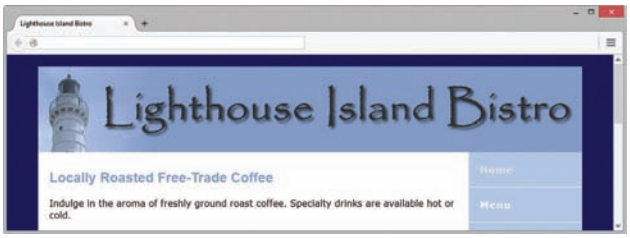

Figure 6.38 The Header Text Image Replacement technique has been applied to the web page

Although the text coded between the h1 tags no longer displays in the browser viewport, it is available to assistive technologies such as screen readers and to search engines. A suggested solution is in the student files chapter6/6.7 folder.

6.11 Practice with an Image Gallery

You have previously used an unordered list to configure a list of navigation hyperlinks. Since a bunch of navigation hyperlinks can be considered a list, the unordered list structure is a semantically correct method to code hyperlinks on a web page. Let's apply this coding method to a gallery of images (which is basically a bunch of images) displayed on a web page in the next Hands-On Practice.

Hands-On Practice 6.8

In this Hands-On Practice you will create the web page shown in Figure 6.39, which displays a group of images with captions.

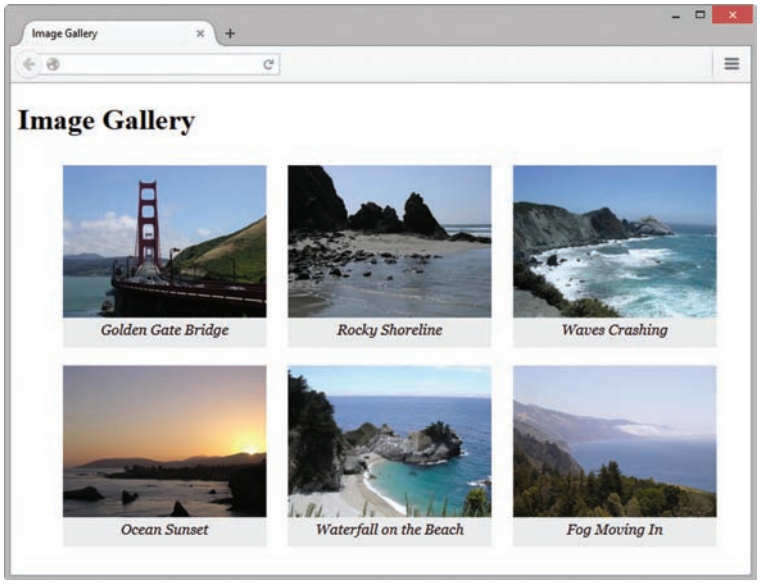

Figure 6.39 The images float in this web page

You'll configure the images and their captions to fill the available space in the browser viewport. The display will change based on the size of the browser viewport.

Figure 6.40 shows the same web page displayed in a browser that has been resized to be narrower.

Create a new folder named gallery. Copy the following images from the student files chapter6/starters folder into the gallery folder: photo1.jpg, photo2.jpg, photo3.jpg, photo4.jpg, photo5.jpg, and photo6.jpg.

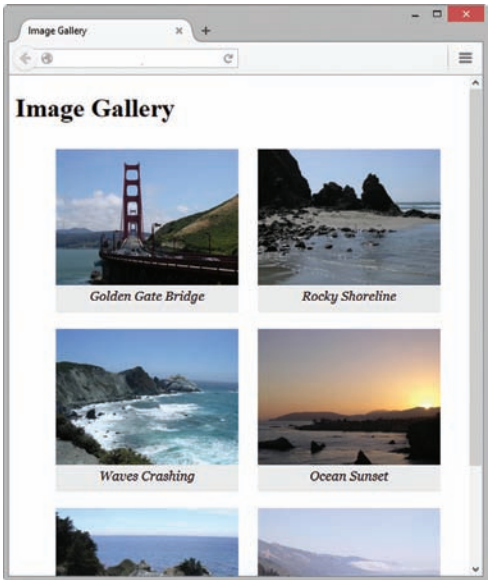

Figure 6.40 The floated images move as the browser is resized

Launch a text editor and open the template file located at chapter2/template.html in the student files. Save the file as index.html in your gallery folder. Modify the file to configure a web page as indicated:

1. Configure the text, Image Gallery, within an h1 element and within the title element.
2. Configure an unordered list. Code six li elements, one for each image along with a brief text caption. An example of the first li element is

```
<li>
<img src="photo1.jpg"
alt="Golden Gate Bridge"
width="225" height="168">
Golden Gate Bridge
</li>
```

3. Configure all six li elements in a similar manner. Substitute the actual name of each image file for the `src` values in the code. Write your own descriptive text for each image. Use photo2.jpg in the second image element, photo3.jpg in the third image element, photo4.jpg in the fourth image element, photo5.jpg in the fifth image element, and photo6.jpg in the sixth image element. Save the file. Display your page in a browser. Figure 6.41 shows a partial screen capture.

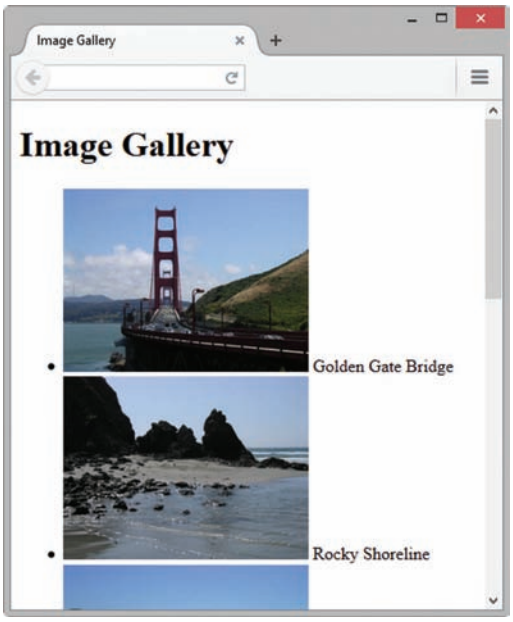

Figure 6.41 The web page before CSS

4. Now, let's add embedded CSS. Open your file in a text editor and code a style element in the head section. Configure the ul element selector to not display a list marker. Configure the li element selector with inline-block display. Also set width to 225px, a 10 pixel margin, 10 pixels of bottom padding, light gray (#EAEAEA) background color, centered text, and italic text in the Georgia (or other serif) font. The CSS follows

```
ul { list-style-type: none; }
li { display: inline-block;
     width: 225px;
     padding-bottom: 10px;
     margin: 10px;
     background-color: #EAEAEA;
     text-align: center;
     font-style: italic;
     font-family: Georgia, serif; }
```

5. Save your page and display it in a browser. Experiment with resizing the browser window to see the display change. Compare your work to Figures 6.39 and 6.40. A sample solution is in the student files (chapter6/6.8).

6.12 Positioning with CSS

You've seen how normal flow causes the browser to render the elements in the order that they appear in the HTML source code. You have also experienced how floating elements can move and shift as the browser viewport is resized. When using CSS for page layout there are situations when you may want more control over the position of an element. The **`position` property** configures the type of positioning used when the browser renders an element. Table 6.6 lists position property values and their purpose.

Table 6.6 The position Property

Value	Purpose
`static`	Default value; the element is rendered in normal flow
`fixed`	Configures the location of an element within the browser viewport; the element does not move when the page is scrolled
`relative`	Configures the location of an element relative to where it would otherwise render in normal flow
`absolute`	Precisely configures the location of an element outside of normal flow

Static Positioning

Static positioning is the default and causes the browser to render an element in normal flow. As you've worked through the exercises in this book, you have created web pages that the browser rendered using normal flow.

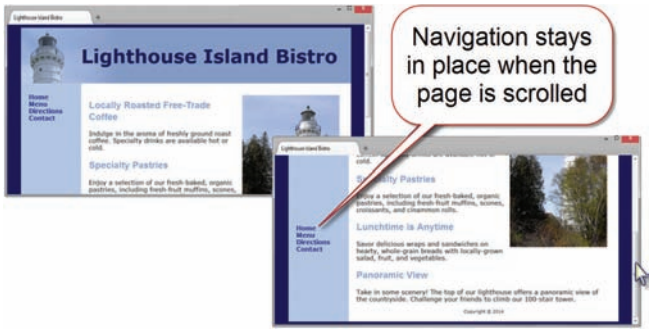

Figure 6.42 The navigation is configured with fixed positioning

Fixed Positioning

Use **fixed positioning** to cause an element to be removed from normal flow and to remain stationary, or "fixed in place," when the web page is scrolled in the browser viewport. Figure 6.42 shows a web page (see chapter6/fixed.html in the student files) with a navigation area configured with fixed position. The navigation stays in place even though the user has scrolled down the page. The CSS follows:

```
nav { position: fixed; }
```

Relative Positioning

Use **relative positioning** to change the location of an element slightly, relative to where it would otherwise appear in normal flow. However, the area in normal flow is still reserved for the element and other elements will flow around that reserved space. Configure relative positioning with the `position: relative;` property along with one or more of the following offset properties: `left, right, top, bottom`. Table 6.7 lists the offset properties.

Table 6.7 The Position Offset Properties

Property	Value	Purpose
`left`	Numeric value or percentage	The position of the element offset from the left side of the container element
`right`	Numeric value or percentage	The position of the element offset from the right side of the container element
`top`	Numeric value or percentage	The position of the element offset from the top of the container element
`bottom`	Numeric value or percentage	The position of the element offset from the bottom of the container element

Figure 6.43 shows a web page (chapter6/relative.html in the student files) that uses relative positioning along with the **`left` property** to configure the placement of an element in relation to the normal flow. In this case, the container element is the body of the web page. The result is that the content of the element is rendered as being offset or shifted by 30 pixels from the left where it would normally be placed at the browser's left margin. Notice also how the background-color and padding properties configure the h1 element. The CSS is

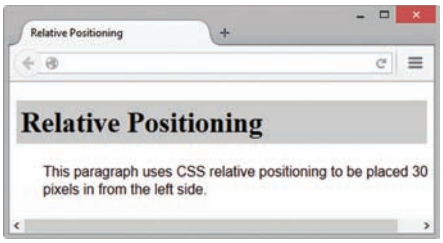

Figure 6.43 The paragraph is configured using relative positioning

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

The HTML source code follows:

```
<h1>Relative Positioning</h1>
<p>This paragraph uses CSS relative positioning to be placed 30 pixels
in from the left side.</p>
```

Absolute Positioning

Use **absolute positioning** to precisely specify the location of an element outside of normal flow in relation to its first parent non-static element. If there is no non-static parent element, the absolute position is specified in relation to the body of the web page. Configure absolute positioning with the `position: absolute;` property along with one or more of the offset properties `(left, right, top, bottom)` listed in Table 6.7.

Figure 6.44 depicts a web page that configures an element with absolute positioning to display the content 200 pixels in from the left margin and 100 pixels down from the top of the web page document. An example is in the student files, (chapter6/absolute.html).

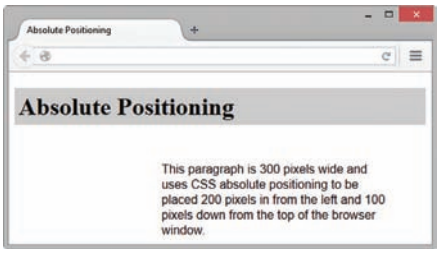

Figure 6.44 The paragraph is configured with absolute positioning.

The CSS is

```
p {  position: absolute;
     left: 200px;
     top: 100px;
     font-family: Arial, sans-serif;
     width: 300px; }
```

The HTML source code is

```
<h1>Absolute Positioning</h1>
   <p>This paragraph is 300 pixels wide and uses CSS absolute
   positioning to be placed 200 pixels in from the left and 100 pixels
   down from the top of the browser window.</p>
```

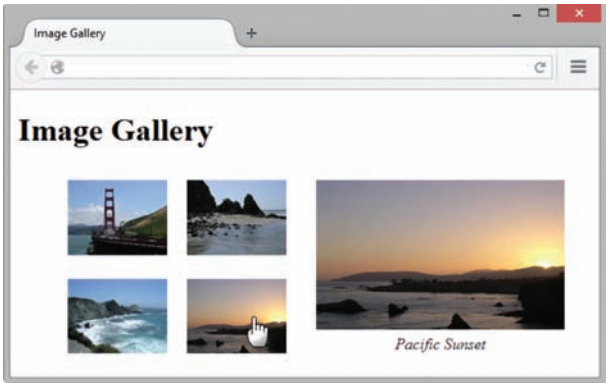

Figure 6.45 An interactive image gallery with CSS.

Practice with Positioning

Recall that the CSS `:hover` pseudo-class provides a way to configure styles to display when the web page visitor moves the mouse over an element. You'll use this basic interactivity along with CSS positioning and display properties to configure an interactive image gallery with CSS and HTML. Figure 6.45 shows the interactive image gallery in action (see chapter6/6.9/gallery.html in the student files). When you place the mouse over a thumbnail image, the larger version of the image is displayed along with a caption. If you click on the thumbnail, the larger version of the image displays in its own browser window.

Hands-On Practice 6.9

In this Hands-On Practice, you will create the interactive image gallery web page shown in Figure 6.45. Copy the following images located in the student files chapter6/starters folder into a folder named gallery2: photo1.jpg, photo2.jpg, photo3.jpg, photo4.jpg, photo1thumb.jpg, photo2thumb.jpg, photo3thumb.jpg, and photo4thumb.jpg.

Launch a text editor and modify the chapter2/template.html file to configure a web page as indicated:

1. Configure the text, Image Gallery, within an h1 element, and within the title element.
2. Code a div assigned to the id named `gallery`. This div will contain the thumbnail images, which will be configured within an unordered list.
3. Configure an unordered list within the div. Code four li elements, one for each thumbnail image. The thumbnail images will function as image links with a `:hover` pseudo-class that causes the larger image to display on the page. We'll make this all happen by configuring an anchor element containing both the thumbnail image

and a span element that comprises the larger image along with descriptive text. An example of the first li element is

```
<li><a href="photo1.jpg"><img src="photo1thumb.jpg" width="100"
   height="75" alt="Golden Gate Bridge">
   <span><img src="photo1.jpg" width="250" height="150"
   alt="Golden Gate Bridge"><br>Golden Gate Bridge</span></a>
</li>
```

4. Configure all four li elements in a similar manner. Substitute the actual name of each image file for the href and src values in the code. Write your own descriptive text for each image. Use photo2.jpg and photo2thumb.jpg in the second li element. Use photo3.jpg and photo3thumb.jpg in the third li element. Use photo4.jpg and photo4thumb.jpg for the fourth li element.
Save the file as index.html in the gallery2 folder. Display your page in a browser. You'll see an unordered list with the thumbnail images, the larger images, and the descriptive text. Figure 6.46 shows a partial screen capture.

Figure 6.46 The web page display before CSS.

5. Now, let's add embedded CSS. Open your index.html file in a text editor and code a style element in the head section. The `gallery` id will use relative positioning instead of the default static positioning. This does not change the location of the gallery but sets the stage to use absolute positioning on the span element in relation to its container (`#gallery`) instead of in relation to the entire web page document. This won't matter too much for our very basic example, but it would be very helpful if the gallery were part of a more complex web page. Configure embedded CSS as follows:

   a. Set the `gallery` id to use relative positioning.
   ```
   #gallery { position: relative; }
   ```
   b. The unordered list in the gallery should have a width of 250 pixels and no list marker.
   ```
   #gallery ul { width: 250px; list-style-type: none; }
   ```
   c. Configure the list item elements in the gallery with inline display, left float, and 10 pixels of padding.
   ```
   #gallery li { display: inline; float: left; padding: 10px; }
   ```
   d. The images in the gallery should not display a border.
   ```
   #gallery img { border-style: none; }
   ```
   e. Configure anchor elements in the gallery to have no underline, #333 text color, and italic text.
   ```
   #gallery a { text-decoration: none; color: #333;
                font-style: italic; }
   ```
   f. Configure span elements in the gallery not to display initially.

```
#gallery span { display: none; }
```

g. Configure the span elements in the gallery to display *only* when the web visitor hovers the mouse over the thumbnail image link. Set the location of the span to use absolute positioning. Locate the span 10 pixels down from the top and 300 pixels in from the left. Center the text within the span:

```
#gallery a:hover span { display: block; position: absolute;
        top: 10px; left: 300px; text-align: center; }
```

Save your page and display it in a browser. Your interactive image gallery should work well in modern browsers. Compare your work to Figure 6.45 and the sample in the student files (chapter6/6.9/gallery.html).

6.13 CSS Debugging Techniques

Using CSS for page layout requires some patience. It takes a while to get used to it. Fixing problems in code is called **debugging**. This term dates back to the early days of programming when an insect (a bug) lodged inside the computer and caused a malfunction. Debugging CSS can be frustrating and requires patience. One of the biggest issues is that even modern browsers implement CSS in slightly different ways. Browser support changes with each new browser version. Testing is crucial. Expect your pages to look slightly different in various browsers. Although Internet Explorer's support of the CSS standard is improving, there still are differences in compliance. The following are helpful techniques to use when your CSS isn't behaving properly:

Verify Correct HTML Syntax

Invalid HTML code can cause issues with CSS. Use the W3C Markup Validation Service at http://validator.w3.org to verify the correct HTML syntax.

Verify Correct CSS Syntax

Sometimes a CSS style does not apply because of a syntax error. Use the W3C CSS Validation Service at http://jigsaw.w3.org/css-validator to verify your CSS syntax. Carefully check your code. Many times, the error is in the line *above* the style that is not correctly applied.

Configure Temporary Background Colors

Sometimes your code is valid but the page is not rendered in the way that 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 quickly.

Use Comments to Find the Unexpected Cascade

Style rules and HTML attributes configured farther down the page can override earlier style rules. 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.

Browsers ignore code and text that is contained between comment markers. A CSS comment begins with `/*` and ends with `*/`. The comment below is an example of documentation that explains the purpose of a style rule.

```
/* Set Page Margins to Zero */
body { margin: 0; }
```

Comments can span multiple lines. The following comment begins on the line above the style declaration for the `new` class and ends on the line below the style declaration for the `new` class. This causes the browser to skip the `new` class when applying the style sheet. This technique can be useful for testing when you are experimenting with a number of properties and may need to temporarily disable a style rule.

```
/* temporarily commented out during testing
.new { font-weight: bold; }
*/
```

A common mistake when using comments is to type the beginning `/*` without subsequently typing `*/` to end the comment. As a result, *everything* after the `/*` is treated as a comment by the browser.

FAQ **Where can I find out more about CSS?**

There are many websites with CSS resources and tutorials, including the following:

- CSS-Tricks: http://css-tricks.com
- Max Design: Two columns with color
  http://www.maxdesign.com.au/articles/two-columns/
- Listamatic: Vertical and horizontal lists with CSS
  http://css.maxdesign.com.au/listamatic
- W3C Cascading Style Sheets: http://www.w3.org/Style/CSS
- HTML5 and CSS3 Browser Support Chart: http://www.findmebyip.com/litmus
- CSS Contents and Browser Compatibility: Peter-Paul Koch's site is dedicated to studying and defeating browser incompatibility related to CSS and JavaScript.
  http://www.quirksmode.org/css/contents.html
- CSS3 Click Chart: http://www.impressivewebs.com/css3-click-chart
- SitePoint CSS Reference: http://reference.sitepoint.com/css

6.14 More HTML5 Structural Elements

You've worked with the HTML5 header, nav, main, and footer elements throughout this book. These HTML5 elements are used along with div and other elements to structure web page documents in a meaningful manner that defines the purpose of the structural areas. In this section you'll explore four more HTML5 elements.

The Section Element

The purpose of a **section element** is to indicate a "section" of a document, such as a chapter or topic. This block display element could contain header, footer, section, article, aside, figure, div, and other elements needed to configure the content.

The Article Element

The **article element** is intended to present an independent entry, such as a blog posting, comment, or e-zine article that could stand on its own. This block display element could contain header, footer, section, aside, figure, div, and other elements needed to configure the content.

The Aside Element

The **aside element** indicates a sidebar or other tangential content. This block display element could contain header, footer, section, aside, figure, div, and other elements needed to configure the content.

The Time Element

The **time element** represents a date or a time. An optional `datetime` attribute can be used to specify a calendar date and/or time in machine-readable format. Use YYYY-MM-DD for a date. Use a 24-hour clock and HH:MM for time. See http://www.w3.org/TR/html-markup/time.html.

Hands-On Practice 6.10

In this Hands-On Practice you'll begin with the two-column Lighthouse Island Bistro home page (shown in Figure 6.30) and apply the section, article, aside, and time elements to create the page with blog postings shown in Figure 6.47.

Create a new folder named ch6blog. Copy the index.html and lighthouselogo.jpg files from the student files chapter6/6.4 folder into the ch6blog folder.

Launch a text editor, and open the index.html file. Examine the source code, and locate the header element.

Figure 6.47 This page utilizes the new elements.

1. Add the tagline "the best coffee on the coast" with a span element within the header element. Your code should look similar to the following

```
<header>
    <h1>Lighthouse Island Bistro</h1>
    <span>the best coffee on the coast</span>
</header>
```

2. Replace the contents of the main element with the following code:

```
<section>
<h2>Bistro Blog</h2>
<aside>Watch for the March Madness Wrap next month!</aside>
  <article>
    <header><h3>Valentine Wrap</h3></header>
    <time datetime="2016-02-01">February 1, 2016</time>
   <p>The February special sandwich is the Valentine Wrap
   &mdash; heart healthy organic chicken with roasted red
   peppers on a whole wheat wrap.</p>
  </article>
  <article>
   <header><h3>New Coffee of the Day Promotion</h3></header>
   <time datetime="2016-01-12">January 12, 2016</time>
   <p>Enjoy the best coffee on the coast in the comfort of your
   home. We will feature a different flavor of our gourmet,
   locally roasted coffee each day with free bistro tastings
   and a discount on one-pound bags.</p>
  </article>
</section>
```

3. Edit the embedded CSS for the h1 element selector. Set the h1 bottom margin to 0.

4. Edit the embedded CSS for the h2 element selector. Set font-size to 200%.
5. Configure embedded CSS for the span element selector. Set 20 pixels of left padding, and .80em italic, #00005D color text.
6. Configure embedded CSS for the header element contained within each article element. Use a descendant HTML selector. Set background color to #FFFFFF, no background image, 90% font size, 0 left padding, and auto height (use `height: auto;`).
7. The aside element contains content that is tangential to the main content. Configure embedded CSS to display the aside element on the right (use float) with a 200 pixel width, light gray background color, 20 pixels of padding, 40 pixels of right margin, 80% font size, and a 5px box shadow. Configure a relative position (use `position: relative; top: -40px;`).

Save your file. Display your index.html page in a browser. It should look similar to the page shown in Figure 6.47. A sample solution is in the student files (chapter6/6.10).

FAQ **Now that the the W3C has approved HTML5 for Recommendation status, will more changes be made?**

HTML5 was placed in Recommendation status by the W3C in late 2014, and is considered to be stable at this time.

However, reaching Recommendation status will not be the end of changes. HTML5 is a living, evolving language. The W3C is already working on HTML5.1—the next version of HTML5! You can review the HTML5.1 draft specification at http://www.w3.org/TR/html51. Expect to see changes as the development of HTML5.1 continues such as new elements and even the removal of elements that were newly added to HTML5.

6.15 HTML5 Compatibility with Older Browsers

Internet Explorer (version 9 and later) and current versions of Safari, Chrome, Firefox, and Opera offer good support for most of the HTML5 elements you've been using. However, not everyone has a recent browser installed on their computer. Some people still use earlier versions of browsers for a variety of reasons. Although this issue will decrease in importance over time as people update their computers, your clients will most likely insist that their web pages are usable to as broad of an audience as possible.

Figure 6.48 Outdated browsers do not support HTML5.

Figure 6.48 shows the web page you created in Hands-On Practice 6.10 displayed in the outdated Internet Explorer 7—it's quite different from the modern display shown in Figure 6.47. The good

news is that there are two easy methods to ensure backward compatibility of HTML5 with older, outdated browsers: configuring block display with CSS and the HTML5 Shim.

Configure CSS Block Display

Add one style rule to your CSS to inform older browsers to display HTML5 elements such as header, main, nav, footer, section, article, figure, figcaption, and aside as block display (with empty space above and below). Example CSS follows:

```
header, main, nav, footer, section, article, figure, figcaption,
aside { display: block; }
```

This technique will work well in all browsers except for Internet Explorer 8 and earlier versions. So, what's to be done about Internet Explorer 8 and earlier versions? That's where the HTML5 Shim (also called the HTML5 Shiv) is useful.

HTML5 Shim

Remy Sharp offers a solution to enhance the support of Internet Explorer 8 and earlier versions (see http://remysharp.com/2009/01/07/html5-enabling-script and https://github.com/aFarkas/html5shiv). The technique uses conditional comments that are only supported by Internet Explorer and are ignored by other browsers. The conditional comments cause Internet Explorer to interpret JavaScript statements (see Chapter 11) that configure it to recognize and process CSS for the new HTML5 element selectors. Sharp has uploaded the script to Google's code project at http://html5shim.googlecode.com/svn/trunk/html5.js and has made it available for anyone to use.

Add the following code to the head section of a web page after CSS to cause Internet Explorer 8 and earlier versions to correctly render your HTML5 code:

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

What's the drawback to this approach? Be aware that your web page visitors using Internet Explorer 8 and earlier versions may see a warning message and must have JavaScript enabled for this method to work.

Hands-On Practice 6.11

In this Hands-On Practice you'll modify the two-column Lighthouse Island Bistro home page (shown in Figure 6.47) to ensure backward compatibility with older browsers. Create a new folder named ch6shim. Copy the following files from the student files chapter6/6.10 folder into the ch6shim folder: index.html and lighthouselogo.jpg.

1. Launch a text editor, and open the index.html file. Examine the source code, and locate the head element and style element.

2. Add the following style declaration to the embedded styles:

```
header, main, nav, footer, section, article, figure, figcaption,
aside { display: block; }
```

3. Add the following code below the closing style tag and above the closing head tag:

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

4. Save your file. Display your index.html page in a modern browser. It should look similar to the page shown in Figure 6.47 in a modern browser. If you have access to a computer with an older version of Internet Explorer (such as IE7), use it to test your page. The solution is in the student files chapter6/6.11 folder.

Visit Modernizr at http://www.modernizr.com to explore a free open-source JavaScript library that enables backward compatibility for HTML5 and CSS3 in older browsers.

Checkpoint 6.2

1. Describe a reason to use HTML5 structural elements instead of div elements for some page areas.
2. Describe one CSS debugging tip that you have found to be helpful.
3. Describe how to choose whether to configure an HTML element selector, create a class, or create an id when working with CSS.

Chapter Summary

This chapter introduced CSS page layout techniques and additional HTML5 structural elements. Techniques for positioning and floating elements and configuring two-column page layouts were demonstrated. The topic of page layout is very deep and you have much to explore. Visit the resources cited in the chapter to continue learning about this technology.

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

Key Terms

`:active`
`:focus`
`:hover`
`:link`
`:visited`
<article>
<aside>
<section>
<time>
absolute positioning
article element
aside element
border
box model
`box-sizing` property
`clear` property
debugging
`display` property
fixed positioning
`float` property
Header Text Image Replacement
`left` property
list markers
`list-style-image` property
`list-style-type` property
margin
`margin` property
normal flow
`overflow` property
padding
`padding` property
`position` property
pseudo-class
relative positioning
section element
static positioning
`text-decoration` property
time element
universal selector
visible width
`width` property

Review Questions

Multiple Choice

1. 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

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

3. Which of the following properties can be used to clear a float?
   a. float or clear
   b. clear or overflow
   c. position or clear
   d. overflow or float

4. Which of the following configures a class called side to float to the left?
   a. `.side { left: float; }`
   b. `.side { float: left; }`
   c. `.side { float-left: 200px; }`
   d. `.side { position: left; }`

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

6. Which of the following is an example of using a descendant selector to configure the anchor tags within the nav element?
   a. `nav. a`
   b. `a nav`
   c. `nav a`
   d. `a#nav`

7. Which property and value are used to configure an unordered list item with a square list marker?
   a. `list-bullet: none;`
   b. `list-style-type: square;`
   c. `list-style-image: square;`
   d. `list-marker: square;`

8. Which of the following causes an element to display as a block of content with white space above and below?
   a. `display: none;`
   b. `block: display;`
   c. `display: block;`
   d. `display: inline;`

9. Which of the following pseudo-classes is the default state for a hyperlink that has been clicked?
   a. `:hover`
   b. `:link`
   c. `:onclick`
   d. `:visited`

10. Which HTML5 element has the purpose of providing tangential content?
    a. article
    b. aside
    c. sidebar
    d. section

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. The _____ pseudo-class can be used to modify the display of a hyperlink when a mouse pointer passes over it.

15. Use the HTML5 _____ element to configure an article or blog post that can stand on its own.

Apply Your Knowledge

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

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>CircleSoft Web Design</title>
<meta charset="utf-8">
<style>
  h1 { border-bottom: 1px groove #333333;
       color: #006600;
       background-color: #cccccc }
  #goal { position: absolute;
          left: 200px;
          top: 75px;
          font-family: Arial, sans-serif;
          width: 300px; }
  nav a { font-weight: bold; }
</style>
```

```
</head>
<body>
<h1>CircleSoft Web Design</h1>
<div id="goal">
<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 website that works.</p>
</div>
<nav>
<ul>
  <li>Home</li>
  <li><a href="about.html">About</a></li>
  <li><a href="services.html">Services</a></li>
</ul>
</nav>
</body>
</html>
```

2. **Fill in the Missing Code.** This web page should be configured as a two-column page layout with a right column (containing the navigation area) that is 150 pixels wide. The right column should have a 1 pixel border. The margin in the left-column main content area needs to allow for space that will be used by the right column. Some CSS selectors, properties, and values, indicated by "_", are missing. Fill in the missing code.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Trillium Media Design</title>
<meta charset="utf-8">
<style>
nav {  "_": "_";
      width: "_";
      background-color: #cccccc;
      border:  "_"; }
header { background-color: #cccccc;
         color: #663333;
         font-size: x-large;
         border-bottom: 1px solid #333333; }
main { margin-right:  "_"; }
footer { font-size: x-small;
         text-align: center;
         clear:  "_"; }
 "_"  a { color: #000066;
        text-decoration: none; }
ul {list-style-type:  "_"; }
</style>
</head>
<body>
<nav>
```

```
<ul>
  <li><a href="index.html">Home</a></li>
  <li><a href="products.html">Products</a></li>
  <li><a href="services.html">Services</a></li>
  <li><a href="about.html">About</a></li>
</ul>
</nav>
<main>
<header>
<h1>Trillium Media Design</h1>
</header>
<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 website that works.</p>
</main>
<footer>
Copyright &copy; 2016 Trillium Media Design<br>
Last Updated on 06/03/16
</footer>
</body>
</html>
```

3. **Find the Error.** When this page is displayed in a browser, the heading information obscures the floating image and paragraph text. Correct the errors and describe the process that you followed.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>CSS Float</title>
<meta charset="utf-8">
<style>
body { width: 500px; }
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>
```

```
<h1>Floating an Image</h1>
<img id="yls" src="yls.jpg" alt="Yellow Lady Slipper" height="100"
width="100">
<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 an id with the following attributes: float to the left of the page, light tan background, Verdana or sans-serif large font, and 20 pixels of padding.
2. 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.
3. 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.
4. 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.
5. Write the CSS for an id with the following characteristics: fixed position, light gray background color, bold font weight, and 10 pixels of padding.
6. Write the CSS to configure an image file named myimage.gif as the list marker in an unordered list.
7. Write the CSS to configure an unordered list to display a square list marker.
8. Configure a web page with a list of hyperlinks to your favorite sites. Use an unordered list without any list markers to organize the hyperlinks. Refer to Chapter 5 for color scheme resources. Choose a background color for the web page and a background color for the following states: unvisited hyperlink, hyperlink with a mouse pointer passing over it, and visited hyperlink. Use embedded CSS to configure the background and text colors. Also use CSS to configure the hyperlink's underline to not display when the mouse pointer is passing over it. Save the file as mylinks.html.
9. Use the mylinks.html file you created in 8 as a starting point. Modify the web page to use external rather than embedded CSS. Save the CSS file as links.css.
10. Create an HTML5 web page about one of your favorite hobbies. Choose a hobby and either take a relevant photo or select a relevant royalty-free photo from the Web (refer to Chapter 4). Decide on a heading for your page. Write one or two brief paragraphs about the hobby. The page must use valid HTML5 syntax and include the following elements: header, article, and footer. Use the figure, figcaption, and img elements to display the photo you have chosen. Include a hyperlink to a website that is relevant to the hobby. Include your name in an e-mail address in the page footer area. Configure the text, color, and layout with embedded CSS. Refer to the section "HTML5 Compatibility with Older Browsers" and review the techniques for configuring HTML5 pages to display in both modern and older versions of browsers. Modify the CSS and HTML of your page for cross-browser display. Save the file as myhobby.html.

Web Research

This chapter introduced using CSS to configure web page layout. Use the resources listed in the textbook 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 (configured as a hyperlink), the name of the website, a brief description, and a rating that indicates how helpful it is to beginning web developers.

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 (configured as a hyperlink), the name of the website, a description of the new technique you discovered, and a discussion of how the technique follows (or does not follow) principles of design.

WEBSITE CASE STUDY

Implementing a CSS Two-Column Page Layout

Each of the following case studies continues throughout most of the textbook. This chapter implements a CSS two-column page layout in the websites.

JavaJam Coffee House

See Chapter 2 for an introduction to the JavaJam Coffee House case study. Figure 2.30 shows a site map for the JavaJam. In this case study, you will implement a new two-column CSS page layout for JavaJam. Figure 6.49 shows a wireframe for a two-column page layout with wrapper, header, navigation, main content, hero image, and footer areas.

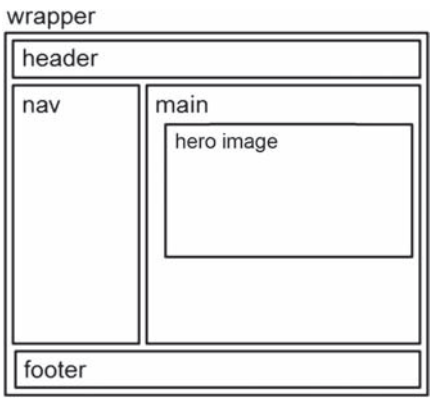

Figure 6.49 Wireframe for a two-column page layout for the JavaJam website

You will modify the external style sheet and the Home, Menu, and Music pages. Use the Chapter 4 JavaJam website as a starting point for this case study. You have five tasks in this case study:

1. Create a new folder for this JavaJam case study.
2. Modify the style rules in the javajam.css file to configure a two-column page layout, as shown in Figure 6.49.
3. Modify the Home page to implement the two-column page layout, as shown in Figure 6.50.

Figure 6.50 The new JavaJam two-column layout (index.html)

4. Modify the Menu page (Figure 6.51) to be consistent with the Home page.
5. Modify the Music page (Figure 6.52) to be consistent with the Home page.

Hands-On Practice Case

Task 1: Create a Folder. Create a folder called javajam6. Copy all of the files from your Chapter 4 javajam4 folder into the javajam6 folder. Copy the heroroad.jpg, heromugs.jpg, and heroguitar.jpg files from the chapter6/starters 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 6.49. See the new JavaJam Home page, as shown in Figure 6.50.

Task 2: Configure the CSS. Open javajam.css in a text editor. Edit the style rules as follows:

1. Configure the universal selector with a `box-sizing: border-box` style declaration.

```
* { box-sizing: border-box; }
```

2. Configure id selectors for the hero image on each page.
   a. Configure an id selector named `heroroad`. Set the background image to heroroad.jpg. Configure 100% background-size and height of 250 pixels.
   b. Configure an id selector named `heromugs`. Set the background image to heromugs.jpg. Configure 100% background-size and height of 250 pixels.
   c. Configure an id selector named `heroguitar`. Set the background image to heroguitar.jpg. Configure 100% background-size and height of 250 pixels.
3. Edit the style rules for the main selector so that the hero image will fill the entire area. Change left padding to 0. Change right padding to 0. Also configure a 200px left margin, 0 top padding, and #FEF6C2 background color.
4. Since the main content area no longer has any left or right padding, configure descendant selectors to configure style rules for the following elements within the main element: h2, h3, h4, p, div, ul, dl. Set left padding to 3em and right padding to 2em.
5. Configure the left-column navigation area. Add style declarations to the nav element selector to configure an area that floats to the left and is 200 pixels wide.
6. Configure the `:link`, `:visited`, and `:hover` pseudo-classes for the navigation hyperlinks. Use the following text colors: #FEF6C2 (unvisited hyperlinks), #D2B48C (visited hyperlinks), and #CC9933 (hyperlinks with `:hover`). For example,
   `nav a:link { color: #FEF6C2; }`
7. You will organize the navigation hyperlinks within an unordered list in later tasks. The navigation area in Figure 6.50 does not show list markers. Code a `nav ul` descendant selector to configure unordered lists in the navigation area to display without list markers and with 0 left padding.
8. Modify the `wrapper` id. Configure a dark background color (#231814) which will display behind the column with the navigation area.
9. Modify the h4 element selector style rules. View the Music page shown in Figure 6.52 and notice that the `<h4>` tags are styled differently, with all uppercase text (use `text-transform`), a bottom border, and 0 bottom padding. Also configure a style declaration to clear floats on the left.
10. Refer to the Music page shown in Figure 6.52 and notice how the images float on the left side of the paragraph description. Configure a new class named `floatleft` that floats to the left with 20 pixels of right and bottom padding.
11. Modify the style rules for the `details` class and add the `overflow: auto;` style declaration.
12. Add the following CSS to be compatible with most older browsers:
    `header, nav, main, footer { display: block; }`

    Save the javajam.css file.

Task 3: The Home Page. Open index.html in a text editor. Edit the code as follows:

1. Add the following HTML5 shim code in the head section of the web page after the link element (to assist Internet Explorer 8 and earlier versions):

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

2. Configure the left-column navigation area, which is contained within the nav element. Remove any ` ` characters that may be present. Code an unordered list to organize the navigation hyperlinks. Each hyperlink should be contained within `<li>` tags.
3. Remove the img tag for the windingroad.jpg image. Configure a div element assigned to the `heroroad` id between the opening main tag and the opening h2 tag. There will be no HTML content in this div, which is a container for a background image you configured with CSS in Task 2.

Save the index.html file. It should look similar to the web page shown in Figure 6.50. Remember that validating your HTML and CSS can help you find syntax errors. Test and correct this page before you continue.

Task 4: The Menu Page. Open menu.html in a text editor. Configure the left-column navigation area, navigation hyperlinks, and HTML5 shim in the same manner as the home page. Remove the img tag for the mugs.jpg image. Configure a div element assigned to the `heromugs` id between the opening main tag and the opening h2 tag. Save your new menu.html page and test it in a browser. It should look similar to the web page shown in Figure 6.51. Use the CSS and HTML validators to help you find syntax errors.

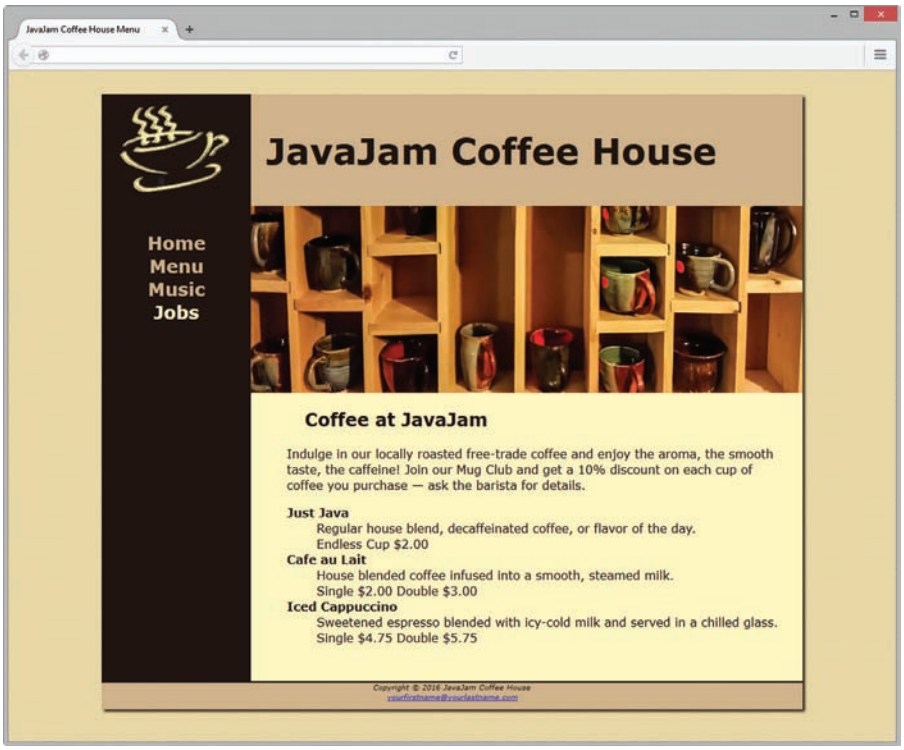

Figure 6.51 The new JavaJam Menu page

Task 5: The Music Page. Open music.html in a text editor. Configure the left-column navigation area, navigation hyperlinks, and HTML5 shim in the same manner as the home page. Configure a div element assigned to the `heroguitar` id between the opening main tag and the opening h2 tag.

Configure the thumbnail images to float to the left. Add `class="floatleft"` to the image tag for each thumbnail image. Save your new music.html page and test it in a browser. It should look similar to the web page shown in Figure 6.52. Use the CSS and HTML validators to help you find syntax errors.

Figure 6.52 The new JavaJam Music Page

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

Fish Creek Animal Hospital

See Chapter 2 for an introduction to the Fish Creek Animal Hospital case study. Figure 2.34 shows a site map for Fish Creek. In this case study, you will implement a redesign with a new two-column CSS page layout. Figure 6.53 displays a wireframe for a two-column page layout with wrapper, header, navigation, main content, and footer areas.

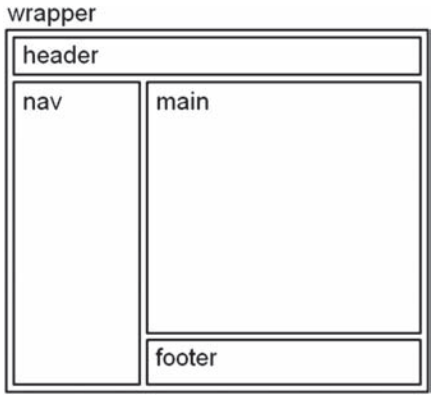

Figure 6.53 Wireframe for a two-column page layout for the Fish Creek website

You will modify the external style sheet and the Home, Services, and Ask the Vet pages. Use the Chapter 4 Fish Creek website as a starting point for this case study. You have five tasks in this case study:

1. Create a new folder for this Fish Creek case study.
2. Modify the style rules in the fishcreek.css file to configure a two-column page layout, as shown in Figure 6.53.
3. Modify the Home page to implement the two-column page layout, as shown in Figure 6.54.
4. Modify the Services page to be consistent with the Home page.
5. Modify the Ask the Vet page to be consistent with the Home page.

Hands-On Practice Case

Task 1: Create a Folder. Create a folder called fishcreek6. Copy all of the files from your Chapter 4 fishcreek4 folder (except the fishcreeklogo.gif image and the fish navigation images home.gif, services.gif, askvet.gif, and contact.gif) into the fishcreek6 folder. Copy the bigfish.gif and gradientblue.jpg images from the chapter6/starters 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 6.53. See the new Fish Creek home page, as shown in Figure 6.54.

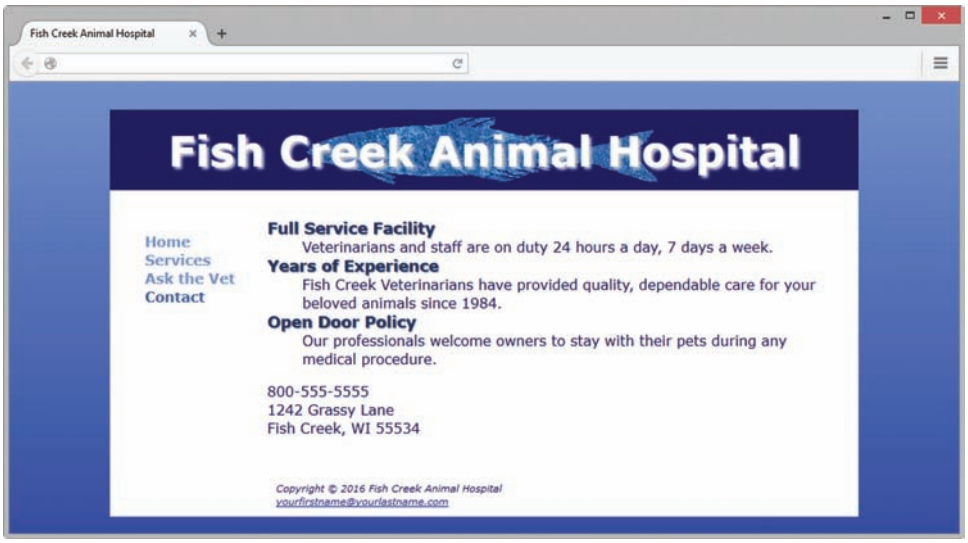

Figure 6.54 The new Fish Creek two-column home page (index.html)

Task 2: Configure the CSS. Open fishcreek.css in a text editor. Edit the style rules as follows:

1. Configure the universal selector with a `box-sizing: border-box` style declaration.
   ```
   * { box-sizing: border-box; }
   ```
2. Configure gradientblue.jpg as a background image for the body element selector.
3. Modify the `wrapper` id. Configure a white background color (#FFFFFF) and dark-blue text color (#000066) for this area.
4. Configure the header element selector. Change the background color to dark-blue (#000066) and the text color to white (#FFFFFF).

Remove the font-family style declaration. In Tasks 3, 4, and 5 you will edit the HTML and replace the fishcreeklogo.gif image with the text, "Fish Creek Animal Hospital". In this step you will configure the bigfish.gif as a background image with center background-position and no repeats.

5. Modify the h1 element selector. Add style declarations for 3em font-size, 10 pixels of padding, 150% line-height, and gray text shadow (#CCCCCC).
6. Configure the left column area. Add new style declarations for the nav element selector to configure an area that floats to the left and is 150 pixels wide. Remove the style declaration for the text-align property.
7. You will organize the navigation hyperlinks within an unordered list in later tasks. The navigation area in Figure 6.54 does not show list markers. Code a `nav ul` descendant selector to configure unordered lists in the navigation area to display without list markers.
8. Configure the navigation anchor tags to display no underline.
9. Configure the `:link`, `:visited`, and `:hover` pseudo-classes for the navigation hyperlinks. Use the following text colors: #3262A3 (unvisited hyperlinks), #6699FF (visited hyperlinks), and #CCCCCC (hyperlinks with `:hover`). For example,
   `nav a:link { color: #3262A3; }`
10. Configure the right column area. Add a new style rule for the main element selector to configure an area with a 180 pixel left margin, 20 pixels of right padding, and 20 pixels of bottom padding.
11. Change the background color configured for the `category` class to #FFFFFF;
12. Configure the footer area. Add style declarations to set 10 pixels of padding and a 180px left margin.
13. Add the following CSS to be compatible with most older browsers:

```
header, nav, main, footer { display: block; }
```

Save the fishcreek.css file.

Task 3: Modify the Home Page. Open index.html in a text editor and modify the code as follows:

1. Add the following HTML5 shim code in the head section of the web page after the link element (to assist Internet Explorer 8 and earlier versions):

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

2. Configure the h1 element. Replace the img tag that displays the fishcreeklogo.gif image with the text, "Fish Creek Animal Hospital".
3. Rework the navigation area. Remove any ` ` characters that may be present. Replace the fish image links with text links. Then, code an unordered list to organize the navigation hyperlinks. Each hyperlink should be contained within `<li>` tags.
4. Remove the nav element and navigation hyperlinks from the footer area.

Save the index.html file. It should look similar to the web page shown in Figure 6.54. Remember that validating your HTML and CSS can help you find syntax errors. Test and correct this page before you continue.

Task 4: Modify the Services Page. Open services.html in a text editor. Configure the h1, navigation area, navigation hyperlinks, footer area, and HTML5 shim in the same manner as the home page. Save your new services.html page and test it in a browser. Use the CSS and HTML validators to help you find syntax errors.

Task 5: Modify the Ask the Vet Page. Open askvet.html in a text editor. Configure the h1, navigation area, navigation hyperlinks, footer area, and HTML5 shim in the same manner as the home page. Save your new askvet.html page and test it in a browser. Use the CSS and HTML validators to help you find syntax errors.

In this case study, you changed the page layout of the Fish Creek website. Notice that with just a few changes in the CSS and HTML code, you configured a two-column page layout with a completely new visual aesthetic.

Pacific Trails Resort

See Chapter 2 for an introduction to the Pacific Trails Resort case study. Figure 2.38 shows a site map for Pacific Trails. The pages were created in earlier chapters. In this case study, you will implement a new two-column CSS page layout. Figure 6.55 displays a wireframe for a two-column page layout with wrapper, header, nav, main content, hero image, and footer areas.

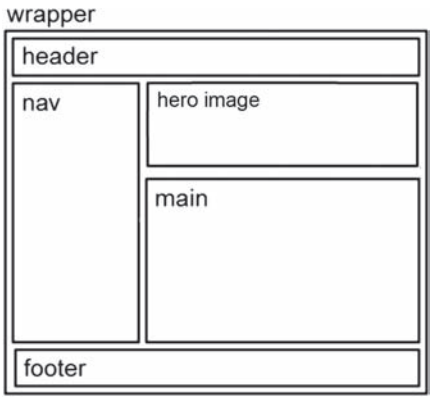

Figure 6.55 Wireframe for a two-column page layout for the Pacific Trails website

You will modify the external style sheet and the Home, Yurts, and Activities pages. Use the Chapter 4 Pacific Trails website as a starting point for this case study. You have five tasks in this case study:

1. Create a new folder for the Pacific Trails case study.
2. Modify the style rules in the pacific.css file to configure a two-column page layout, as shown in Figure 6.55.
3. Modify the Home page to implement the two-column page layout, as shown in Figure 6.56.
4. Modify the Yurts page to be consistent with the Home page.
5. Modify the Activities page to be consistent with the Home page.

Hands-On Practice Case

Task 1: Create a Folder. Create a folder called pacific6. Copy all of the files from your Chapter 4 pacific4 folder into the pacific6 folder. You will modify the pacific.css file and each web page file (index.html, yurts.html, and activities.html) to implement the two-column page layout, as shown in Figure 6.55. See the new Pacific Trails home page, as shown in Figure 6.56.

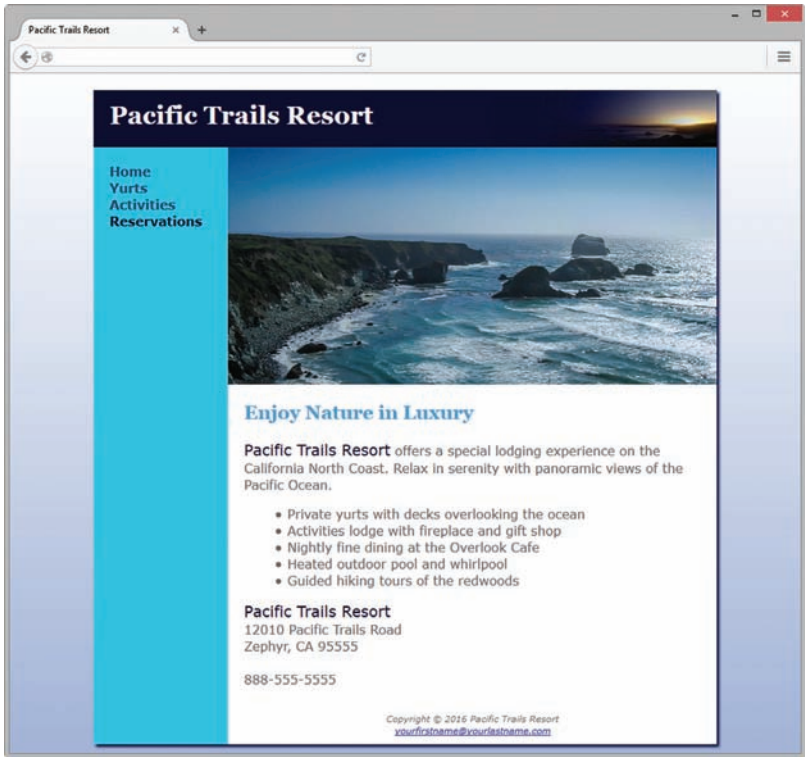

Figure 6.56 The new Pacific Trails two-column home page (index.html)

Task 2: Configure the CSS. Open pacific.css in a text editor. Edit the style rules as follows:

1. Configure the universal selector with a `box-sizing: border-box` style declaration.

   ```
   * { box-sizing: border-box; }
   ```

2. Modify the `wrapper` id. Change the background color to blue (#90C7E3), which will be the background behind the navigation area.

3. Configure the left-column navigation area. Modify the styles for the nav element selector. Keep the style declaration that configures bold text. Remove the background color declaration. The nav area will inherit the background color of the `wrapper` id. Add style declarations to configure this area to float to the left with a width of 160 pixels. Also configure 20 pixels top padding, 5 pixels right padding, no bottom padding, and 20 pixels left padding.

4. Configure the `:link`, `:visited`, and `:hover` pseudo-classes for the navigation hyperlinks. Use the following text colors: #000033 (unvisited hyperlinks), #344873 (visited hyperlinks), and #FFFFFF (hyperlinks with `:hover`). For example,

   ```
   nav a:link { color: #000033; }
   ```

5. You will organize the navigation hyperlinks within an unordered list in later tasks. The navigation area in Figure 6.56 does not show list markers. Code a `nav ul` descendant selector to configure unordered lists in the navigation area to display without list markers. Also configure the unordered list to have no margin and no left padding.

6. Configure the right-column main content area. Modify the styles for the main element selector. Add style declarations to configure a white (#FFFFFF) background, 170 pixels of left margin, 1 pixel of top padding, and 1 pixel of bottom padding.

7. Configure each hero image area (`#homehero`, `#yurthero`, and `#trailhero`) with a 170 pixel left margin.

8. Configure the footer area. Add style declarations to set a white (#FFFFFF) background color and a 170px left margin.
9. Add the following CSS to be compatible with most older browsers:

```
header, nav, main, footer { display: block; }
```

Save the pacific.css file.

Task 3: Modify the Home Page. Open index.html in a text editor and modify the code as follows:

1. Add the following HTML5 shim code in the head section of the web page after the link element (to assist Internet Explorer 8 and earlier versions):

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

2. Configure the left-column navigation area, which is contained within the nav element. Remove any ` ` characters that may be present. Code an unordered list to organize the navigation hyperlinks. Each hyperlink should be contained within `<li>` tags.

Save the index.html file. It should look similar to the web page shown in Figure 6.56. Remember that validating your HTML and CSS can help you find syntax errors. Test and correct this page before you continue.

Task 4: Modify the Yurts Page. Open yurts.html in a text editor. Configure the left-column navigation area, navigation hyperlinks, and HTML5 shim in the same manner as the home page. Save your new yurts.html page and test it in a browser. Use the CSS and HTML validators to help you find syntax errors.

Task 5: Modify the Activities Page. Open activities.html in a text editor. Configure the left-column navigation area, navigation hyperlinks, and HTML5 shim in the same manner as the home page. Save your new activities.html page and test it in a browser. Use the CSS and HTML validators to help you find syntax errors.

In this case study, you changed the page layout of the Pacific Trails Resort website. Notice that with just a few changes in the CSS and HTML code, you configured a two-column page layout.

Path of Light Yoga Studio

See Chapter 2 for an introduction to the Path of Light Yoga Studio case study. Figure 2.42 shows a site map for the Path of Light Yoga Studio. In this case study, you will implement a new two-column CSS page layout for the Path of Light Yoga Studio. Figure 6.57 displays a wireframe for a two-column page layout with a wrapper, header, navigation, main content, and footer area.

Figure 6.57 Wireframe for a two-column page layout for the Path of Light Yoga Studio website

You will modify the external style sheet and the Home, Classes, and Schedule pages. Use the Chapter 4 Path of Light Yoga Studio website as a starting point for this case study. You have five tasks in this case study:

1. Create a new folder for the Path of Light Yoga Studio case study.
2. Modify the style rules in the yoga.css file to configure a two-column page layout, as shown in Figure 6.57.
3. Modify the Home page to implement the two-column page layout, as shown in Figure 6.58.
4. Modify the Classes page to to implement the two-column page layout, as shown in Figure 6.59.
5. Modify the Schedule page to be consistent with the Classes page.

Hands-On Practice Case

Task 1: Create a Folder. Create a folder called yoga6. Copy all of the files from your Chapter 4 yoga4 folder into the yoga6 folder. Copy the yogadoor2.jpg file from the chapter6/starters folder into your yoga6 folder. You will modify the yoga.css file and each web page file (index.html, classes.html, and schedule.html) to implement the two-column page layout shown in Figure 6.57. See the new Path of Light Yoga Studio home page in Figure 6.58.

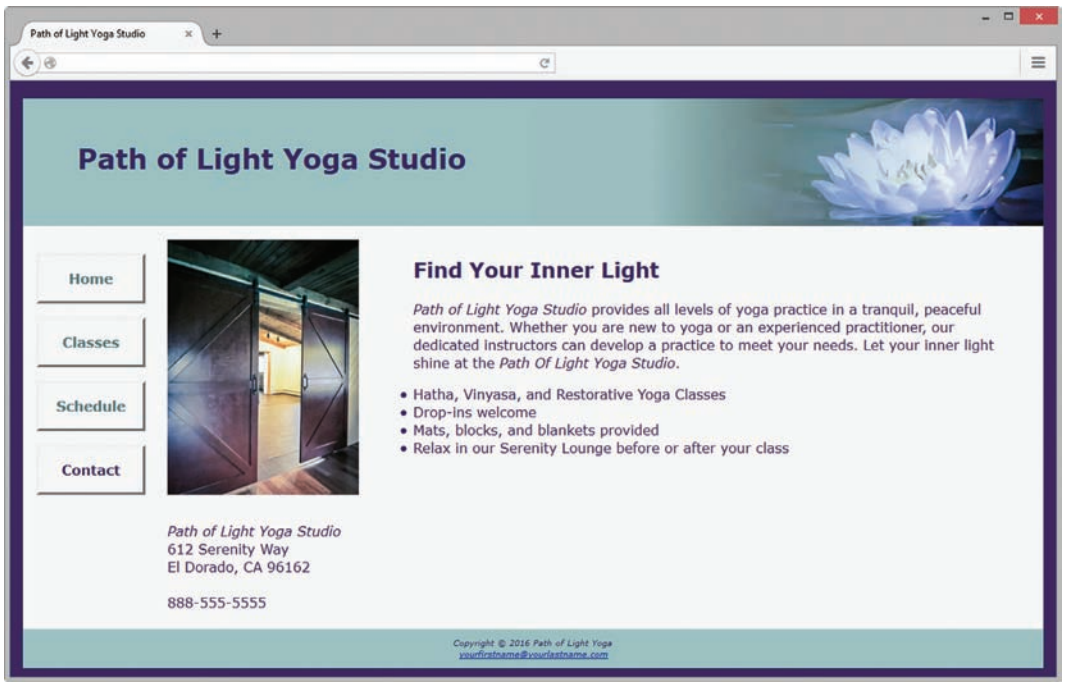

Figure 6.58 The new Path of Light Yoga Studio two-column home page (index.html)

Task 2: Configure the CSS. Open yoga.css in a text editor. Edit the style rules as follows:

1. Configure the universal selector with a `box-sizing: border-box` style declaration.
   ```
   * { box-sizing: border-box; }
   ```
2. Edit the styles for the `wrapper` id. Change min-width to 1200px. Change max-width to 1480px.

3. Configure the left-column navigation area. Modify the styles for the nav element selector. Keep the style declarations that configure bold text and padding. Remove the text-align declaration. The nav area will inherit the background color of the `wrapper` id. Add style declarations to configure this area to float to the left with a width of 160 pixels.
4. Configure the navigation hyperlinks to look like buttons. We'll set up the CSS in this step.
   a. Edit the styles for the `nav a` selector. Keep the text-decoration style declaration. Also configure styles to use block display, centered text, bold font, a 3 pixel gray (#CCCCCC) outset border, 1em padding, and a 1em bottom margin.
   b. Configure the `:link`, `:visited`, and `:hover` pseudo-classes for the navigation hyperlinks. Use the following text colors: #3F2860 (unvisited hyperlinks), #497777 (visited hyperlinks), and #A26100 (hover). Also configure a 3 pixel inset #333333 border for hyperlinks in the hover state.

```
nav a:link { color: #3F2860; }
nav a:visited { color: #497777; }
nav a:hover { color: #A26100; border: 3px inset #333333; }
```

5. You will organize the navigation hyperlinks within an unordered list in later tasks. The navigation area in Figure 6.58 does not show list markers. Code a `nav ul` descendant selector to configure unordered lists in the navigation area to display without list markers. Also configure the unordered list to have no left padding.
6. Edit the styles for the main element selector. Add new style declarations to configure a 170 pixel left margin and 1em top padding.
7. Remove the img element selector and style declarations.
8. Configure a new class named `floatleft` that floats to the left with right margin set to 4em.
9. Remove the `#hero` selector and style declaration.
10. Configure styles for a new class named `clear` with a `clear: both;` style declaration.
11. Add the following CSS to be compatible with most older browsers:

```
header, nav, main, footer { display: block; }
```

Save the yoga.css file.

Task 3: Modify the Home Page. Open index.html in a text editor and modify the code as follows:

1. Add the following HTML5 shim code in the head section of the web page after the link element (to assist Internet Explorer 8 and earlier versions):

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

2. Rework the navigation area. Remove any ` ` characters that may be present. Configure an unordered list to organize the navigation hyperlinks. Each hyperlink should be contained within `<li>` tags.

3. Edit the img tag. Remove the `align="right"` attribute. Assign the img tag to the class named `floatleft`. Change the value of the src attribute to yogadoor2.jpg.
4. Edit the div element that contains the address information. Assign the div to the class named `clear`.

Save the index.html file. It should look similar to the web page shown in Figure 6.58. Remember that validating your HTML and CSS can help you find syntax errors. Test and correct this page before you continue.

Task 4: Modify the Classes Page. Open classes.html in a text editor. Configure the left-column navigation area, navigation hyperlinks, and HTML5 shim in the same manner as the home page. Save your new classes.html page and test it in a browser. It should look similar to Figure 6.59. Use the CSS and HTML validators to help you find syntax errors.

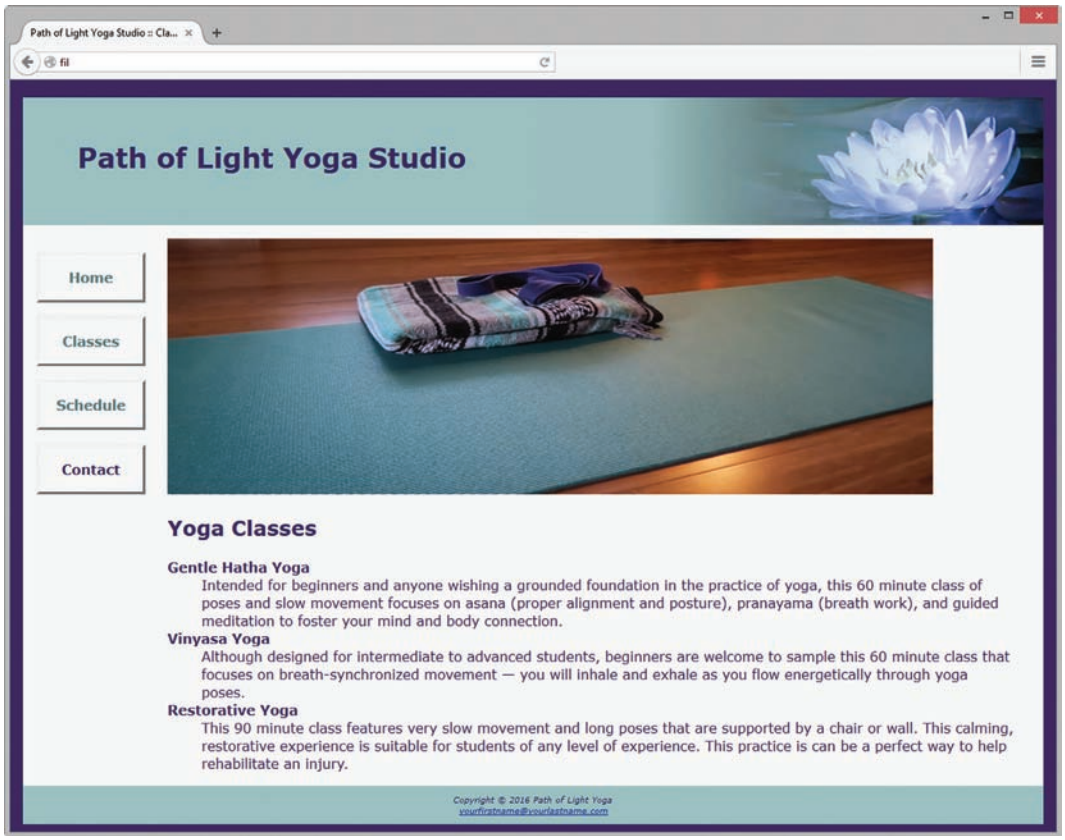

Figure 6.59 The new Path of Light Yoga Studio two-column Classes page

Task 5: Modify the Schedule Page. Open schedule.html in a text editor. Configure the left-column navigation area, navigation hyperlinks, and HTML5 shim in the same manner as the home page. Save your new schedule.html page and test it in a browser. It should look similar to Figure 6.59. Use the CSS and HTML validators to help you find syntax errors.

In this case study, you changed the page layout of the Path of Light Yoga Studio website. Notice that with just a few changes in the CSS and HTML code, you configured a two-column page layout.

Web Project

See Chapter 5 for an introduction to the Web Project case study. 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 in an external style sheet for both formatting and page layout.

Hands-On Practice Case

1. Create a folder called project. All of your project files and graphics will be organized in this folder and in 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 header, navigation, footer, and so on, and list any special configurations needed for these areas. These will be configured 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. 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 them 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.