CHAPTER 10

BUILDING WEBSITES

Throughout this book, you have learned the fundamentals of web development using HTML, XHTML, and CSS. These technologies provide a robust development environment designed to support the creation of web sites. This final chapter is designed to tie together everything you have learned by guiding you through the development of a new website. Along the way, you will learn the basic design steps involved in the development of new sites. By the time you are done, you will be ready to start creating websites of your own.

Specifically, you will learn:

- Fundamental steps and principles involved in website development
- The importance of proper planning and content organization prior to web development
- How to develop an effective navigation system to facilitate movement between the site's web pages
- How to apply a consistent look and feel to all your web pages

PROJECT PREVIEW: WWW.TECH-PUBLISHING.COM

In the previous chapters, you learned how to work with both HTML and XHTML and to use CSS to develop style rules that influence how the resulting markup is rendered. As this book's final project, you will learn how to develop an entire

website. The premise behind the development of this website is that its owner, an author of a number of computer books, wants to create a new website designed to provide additional value and support to his readers.

The website you'll develop in this chapter can be found online at www.tech-publishing.com. There is one difference from the website presented here and the one you will find online: The online version of this website's library and downloads have been modified to use Ajax to supply some of their content, dynamically using data supplied by a web server. Ajax, which stands for Asynchronous JavaScript and XML, is a collection of web development technologies that uses (X)HTML, CSS, JavaScript, the DOM, the XMLHttpRequest object, and XML to create web applications that provide dynamic content provided by a web server.

If you really want to be a great web developer, learning Ajax is a must and is a logical next step to this book. To learn more, check out *Ajax Programming for the Absolute Beginner* (ISBN: 1598635646).

This website will consist of four document styles using a single CSS style sheet. The main landing page for this website is its welcome page, shown in Figure 10.1.

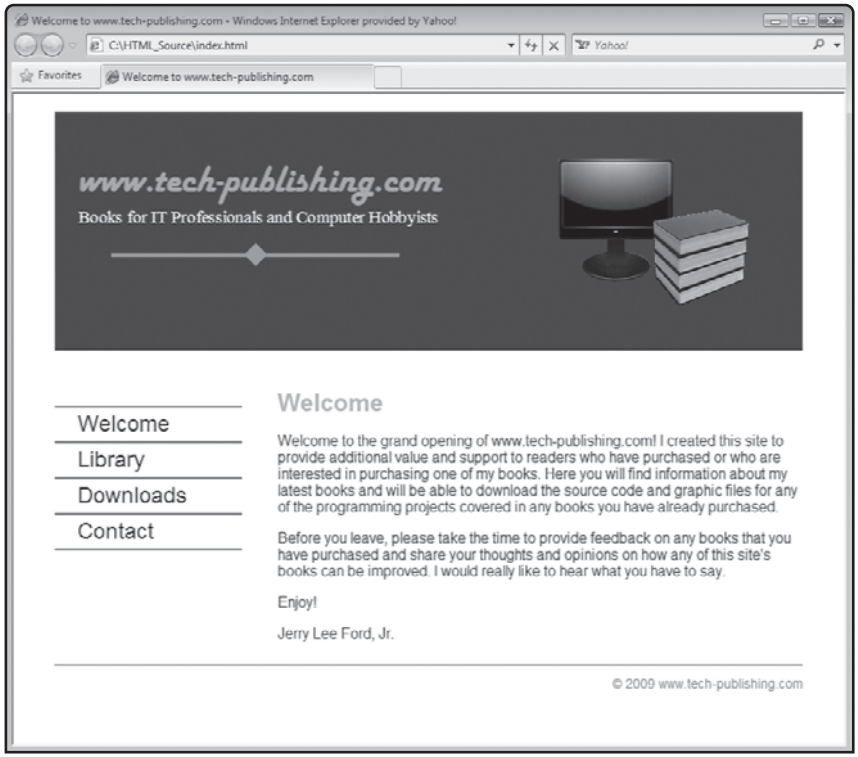

Figure 10.1

The other pages that make up this website are accessed via a series of links that make up the site menu system.

The website's library page, shown in Figure 10.2, provides information on the author's latest books and displays a link to amazon.com where the books can be purchased.

FIGURE 10.2
Every page on the website features the same banner, menu, and font/color scheme.

The Downloads page, shown in Figure 10.3, provides easy access to the source code and graphic files belonging to the author's books. Access to these files is provided by a series of links stored as an unordered list.

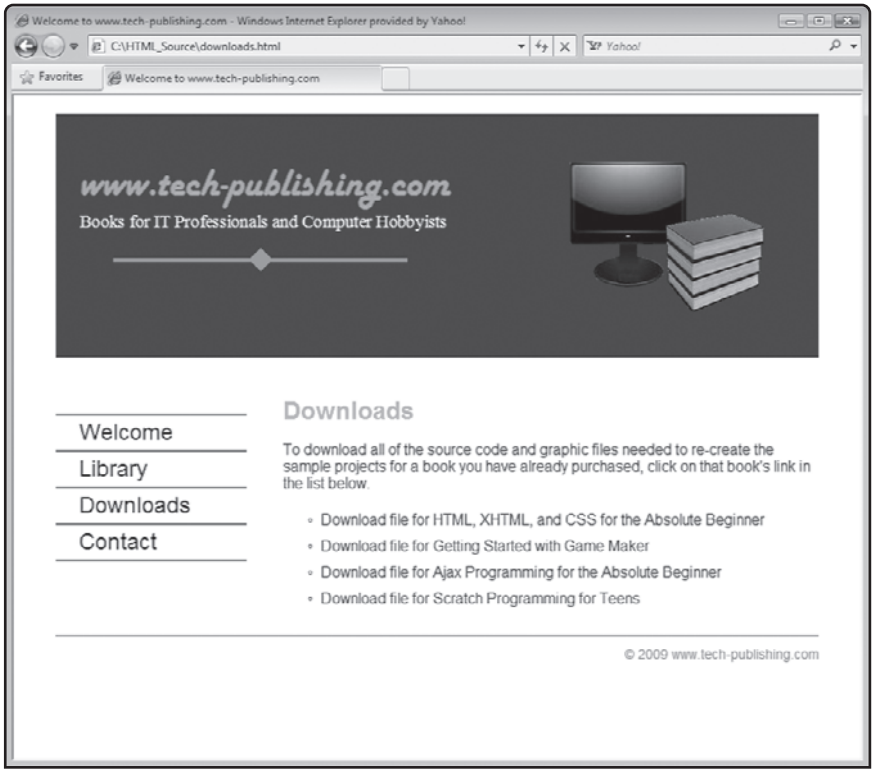

FIGURE 10.3

An unordered list provides single-click access to files containing each book's source code and graphics.

A copyright statement is provided at the bottom of every page in the lower-right corner, immediately after an orange-colored graphic separator bar. The website's final document generates its Contact page, shown in Figure 10.4. This page displays a form that visitors can fill in to provide feedback to the author.

Figure 10.4
A form provides a mechanism for readers to interact with the author.

Designing a Website from the Ground Up

It is important to realize from the very start that there is no specific procedure that must be followed when designing and developing new websites. However, in general it is a good idea to spend a little time thinking about what you hope to accomplish and then figuring out how you want to go about it before you start working. Failure to do so can lead to frustration and a lot of wasted time.

Document Project Objectives

When starting any new development project, a good first step is to begin by documenting your objectives for the project. What information are you trying to share or what products or services are you attempting to sell? Take a look at any competitive websites that might already serve the same population you are targeting. Once you have documented this information, you will be able to use it to help guide any development decisions as you work on your project.

Organization Content

Once you have a good understanding of your objectives, you can begin thinking about how you should lay out the website's overall organization and structure. A good way of doing this is to create a graphic structure chart like the one shown in Figure 10.5.

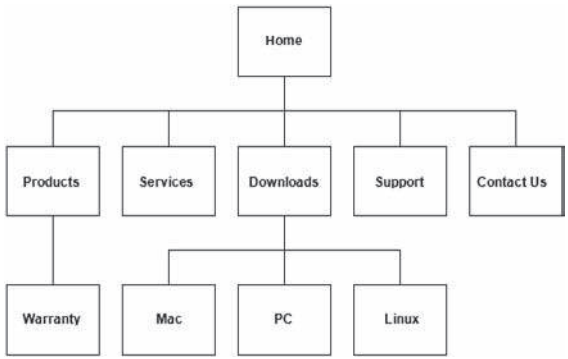

FIGURE 10.5
An example of a website's layout documented using a structure chart.

As Figure 10.5 shows, it is important at this stage to determine the different types of content you plan to provide and to group that content into sections (web documents). By laying out the website's content using a structure chart, you can accomplish multiple objectives. For starters, you will end up with a structure that can be used to layout the design of all of the web documents you will need to create as part of the website. In addition, the structure chart helps define how these documents relate to one another, showing their interconnections and providing a basis for formulating navigation links or menus for your website.

For example, using the structure chart shown in Figure 10.5, you can see that the website has a landing page where all visitors are expected to enter, though thanks to bookmarks and search engine results, this may not always be the case. The landing page provides direct access to five other pages. Your website's navigation links should most likely be based on these five pages. Pages that are connected to one of the individual pages can be accessed by links on those individual pages. However, regardless of where visitors are, it is almost always a good idea to make sure that the website's navigation links or menu is made available on every page, making it easy for visitors to get back to the website's primary pages.

All websites have an initial landing page, intended to be used as the entry point for the people who visit. When you sign up with a web service provider, be sure to find out the name you must assign to your web page's landing page. Usually this file is named index.html. However, in some cases your web service provider may require you to name it default.html, home.html, or something similar.

Outlining a Common Page Structure

In order to ensure that all of the web documents that make up a website share a common look and feel, many web developers begin web development by outlining all of the features that you see as being common to each web document. Once outlined, this list can be used as the basis for creating a template document that outlines the overall layout of the site's web pages. For example, many web pages display a heading at the top with a series of menu links following under it or along the left-hand side of the browser window. In addition, a large area of space is usually reserved for displaying content. Many websites also display a copyright message at the bottom of the window.

Creating a Rough Mockup of the Web Page Template

A good next step at this point is to spend a little time sketching out your idea as to how the website should look. If you have more than one idea, then sketch them all out and then choose whichever one looks best. Assuming that you have created a common page structure layout as previously suggested, you can use it as the basis for ensuring that your page sketch addresses all of the basic requirements.

If you are being paid to build this web page as a part of your job then you can share your sketch with your boss or customer and ask them to review and approve it. This way you will not end up wasting any time and energy developing a website that nobody wants. Your visual outline can be as simple as a hand-drawn sketch or you may instead want to use your favorite graphics program to render a more detailed and realistic example.

Creating a Common Document Template

Assuming that your sketch meets the needs of boss or customer, or you own personal requirements, then you can proceed with the process of constructing a template document. The markup for this document should define all of the content previously outlined. As you lay out the markup for this document, make sure that you keep your focus on defining content based on the order of its importance and not based on the order in which you want it to appear in the browser window. Right now, what you want to do is give your markup a good well-reasoned structure. You can use CSS, as explained in the next step, to reposition content.

Developing a Common CSS Style Sheet for the Website

Once your website's common template has been constructed, you should start working on the website's style rules, preferably using an external style sheet. Make sure that you style all of the content located in the template document so that it is rendered the way you want it to be. If necessary, you can always come back and tweak the CSS style sheet as needed as you work through the development of the entire website. For example, you may find that you

need to come back and add additional CSS rules to fine-tune the presentation of content of some of the individual documents that you will ultimately be creating.

Build-out the Documents That Make Up the Website

Once your common template has the look you want, you can use copies of it as the starting point for each page you plan to add to the website. Assuming that you have carefully worked through the entire development process as just described, then you should find that the development of individual web documents goes pretty smoothly and that when you are done, each resulting web page is rendered with a presentation and style that makes it feel like it's part of a greater whole.

Back to the www.tech-publishing.com Website

It is time to turn your attention back to this book's final project, the creation of the www.tech-publishing.com website. The development of this website will follow the development process previously outlined in this chapter, demonstrating the effectiveness of this common development approach. However, there is more than one way to skin a cat, and you are of course free to follow any development approach you want.

Designing the Website

To keep things as simple and straightforward as possible, this website will be developed by following a specific series of steps, as outlined here:

1. Outline site objectives.
2. Sketch the site's overall structure.
3. Outline template content.
4. Sketch web page design.
5. Create the template markup.
6. Develop the site's CSS file.
7. Assemble document files.
8. Test your new website.

Step 1: Outlining Objectives for the Website

In keeping with the overall methodology outlined earlier in this chapter, let's begin this development effort by outlining the objective for this website. First, it has been stated that its overall purpose is to assist the author in providing additional value and support to his readers. To this end, the website needs to provide the author with a place to talk about his books. Second, since many of the author's books include sample code and graphics, the website needs to make it easy for the reader to locate and download these files. Lastly, the website

should facilitate communication with visitors and provide them with the ability to send messages to the author.

Step 2: Sketching Out the Site's Structure

With these objectives now outlined, it is time to sit down and sketch out an overall outline for the website. For starters, like most websites, this site's general expectation is that visitors will arrive at the website most of the time through its main landing page (although some visitors may arrive at other pages thanks to previously set bookmarks or through search results).

Given the three objectives outlined in the previous step, it makes good sense to organize the rest of the website into three web documents, one to provide information about books, one to allow for source file downloads, and one to provide the reader a means of contacting the author. Figure 10.6 provides a high-level sketch of the website that follows the outline just provided.

FIGURE 10.6
This www.tech-publishing.com website consists of four documents.

Step 3: Outlining Template Content

Now that we know the objectives and have laid out a sketch of the overall organization of the website, it is time to outline the content that should be provided on each of the site's web pages. For starters, let's brand the website by giving it a distinctive banner that is displayed across the top of each web page. Using graphics supplied by the author, display the www.tech-publishing.com logo as well as a graphic showing a computer and a stack of books on the banner.

Some type of navigation controls are required in order to control navigation between the pages that make up the website. To this end, let's develop a simple, text-based navigation menu with links to each of the site's four pages. Of course, each page has specific content that needs to be provided. To facilitate this, a specific portion of the browser window needs to be set aside. To help provide a consistent look and feel, this content area should always be set up to display a heading followed by a paragraph, after which additional content can be displayed.

Lastly, like most websites, a copyright notice needs to be displayed somewhere on the browser window, protecting the author's intellectual property rights. To make all of this information easier to digest, let's rework it as a bulleted list as shown here.

- Solid gray banner along the top of the browser window
  - www.tech-publishing.com graphic logo
  - Graphic image of a computer and books
- A text-based navigation menu
  - Links to each of the site's web documents
- A content area
  - A heading that identifies page content
  - A paragraph for displaying content
- A copyright notice

Now that we have this list of contents, let's use it as the basis for sketching out the overall design of all of the site's web documents and for developing the web page's markup.

Step 4: Sketching Out a Web Page Design

Rather than simply beginning the development of all of the website's documents, it's a good idea to pause and sketch out your vision as to how you plan to lay out the presentation of the web pages. In the case of the www.tech-publishing.com website, the sketch depicted in Figure 10.7 provides a vision of how the final result will look.

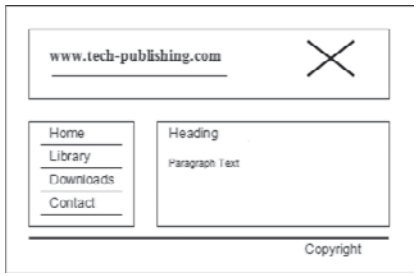

Figure 10.7
Each web page will feature a header, menu link, and a content area.

As you can see, the vision here is to display a heading that spans across the top of the browser window. The header will display the www.tech-publishing.com logo on the left-hand side as well as a graphic showing a computer and stack of books, represented by a bold X on the right-hand side of the header. A text-based collection of links will be displayed on the left-hand side of the browser window to provide navigation controls for every page that makes up the website. To the right of the menu links is the main display area for the web page, where the real content for each page will be displayed.

Beneath all of this, at the bottom of the web page is a graphic separator followed by a copyright notice located at the lower-right corner. The color scheme used to develop the website is orange on dark gray. The www.tech-publishing.com logo, menu headings, and the separate bar will be displayed in orange. The graphic header background color is dark gray. All text displaying in the menu and content area is displayed in black. However, when the mouse pointer is moved over or clicks on a text menu, its color changes to orange.

With this sketch now complete, you can approach the author and show him your idea for how the website will look and feel. Assuming he approves, you can begin working on the website's markup, confident that you are moving in the right direction. If, on the other hand, the author disapproves, you can make changes to your plans without losing a lot of time and effort.

Consider creating a number of different sketches outlining different presentation options for the website and giving all of them to your customer to consider. For example, you might create a sketch to show the page's links listed horizontally across the top of the web page, just under its header. Or you might sketch out a page that uses different types of graphic navigation buttons. The idea here is get the customer to agree to the content being provided and then to a number of different presentation options. Once the customer has made a selection, you can use it as the basis for developing the web page's external CSS file.

Step 5: Creating Template Markup

Once the author has given his approval to your plans, it is time to get to work. Begin by creating a template page that contains all of the content elements outlined in step 3. Don't worry about presentation at this point. All that matters is that you lay out the template document content in a logical and well-structured manner. Name your template document template.html, add the following content to it, and save it.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

  <head>
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8" />
    <title>Welcome to www.tech-publishing.com</title>
    <link rel = "stylesheet" type = "text/css" href = "tech-pub.css" />
  </head>

  <body>
```

```
    <div id = "logo">
      <img src = "banner.png" width = "426" height = "192"
        alt = "Books for Professional and Computer Hobbyists" />
    </div>

    <div id = "concept">
      <img src = "pcandbooks.png" width = "210" height = "160"
        alt = "Picture of a computer and books" />
    </div>

    <ul id = "menu">
      <li><a href="index.html">Welcome</a></li>
      <li><a href="library.html">Library</a></li>
      <li><a href="downloads.html">Downloads</a></li>
      <li><a href="contact.html">Contact</a></li>
    </ul>

    <div id = "content">
      <h1>Page Title</h1>
      <p>Page content.</p>
    </div>

    <div id = "copyright">
      &copy; 2009 www.tech-publishing.com
    </div>

  </body>

</html>
```

As you can see, the `head` section already contains a link element that references a CSS file named tech-pub.css. Even though you have not created this file yet, it is a good idea to link to it now. This will not cause any errors and will ensure that the element is already in place when you later use this template as the basis of creating the site's documents.

The page's `body` section uses a series of `div` elements, each of which is assigned a unique `ID` attribute. The first two `div` elements each enclose an `img` element. The first image element references an image file named banner.png, which contains a graphic showing the website's graphic logo. The second `div` element references a graphic file named pcandbooks.png, which contains a graphic image of a computer and a stack of books.

The next `div` element contains an unordered list, made up of four links that when clicked, load each of the website's pages. Another `div` element, assigned an `ID` of content, has been added to define the area on the document where its content can be assigned. Lastly, a final `div` element is used to display a text string representing the website's copyright message.

Note the inclusion of `&copy;` at the beginning of the copyright message. `&copy;` is an example of a common character reference. It is necessary to use common character references to generate the display of certain types of characters like the copyright (©) character. For a complete list of common character references, check out http://www.w3schools.com/tags/ref_entities.asp.

Right now, the focus of this step is on laying out the markup for template.html in a manner that is well structured. Presentation of content is not important at the moment. Presentation will be addressed in the next step when you develop the document's external style sheet. Figure 10.8 shows how the browser renders the content of this document when loaded.

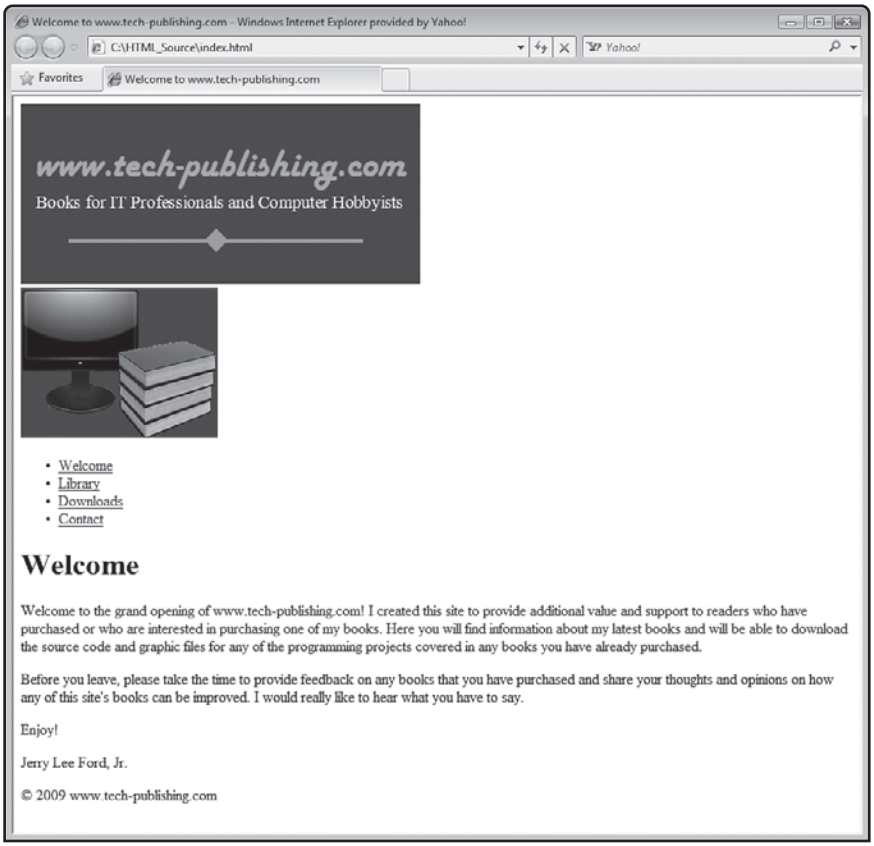

FIGURE 10.8

An example of how the resulting web page looks when the document is rendered by the browser using default style rules.

Step 6: Developing the Site's External CSS File

Now that the markup for the template.html page has been developed, it is time to create the style rules needed to modify the presentation of the resulting web page so that it looks like the sketch that was approved by the author. Begin by creating a CSS file named tech-pub.css and then add the following rule to it.

```
/*Configure minimum page width, available width, margins and default*/
/*font family for the document*/
body {
  min-width: 800px;
  width: 90%;
  margin: 0 auto;
  padding: 0;
  font-family: Arial, sans-serif;
}
```

This rule sets a minimum width of 800 pixels for the web page. It then sets the available area within this space to 90%. Browsers have a tendency to add a little extra padding to elements that make precise control over the presentation of content challenging. To overcome this problem, you simply need to remove any extra margin or padding that may be automatically added by the browser, which you can do by setting the `margin` and `padding` properties to 0. Lastly, `Arial` is set up as the default font for the entire web page.

Next, let's configure the colors used to display the document's links. To do so, add the following style rules to the end of the CSS file.

```
/*Configure the colors when displaying links*/
a:link, a:visited {
  color: black;
}
a:hover, a:focus, a:active  {
  color: orange;
}
```

The first rule configures the display of all links as black. The second rule turns the links to orange when the user moves the mouse pointer over a link or uses the Tab key to select a link. The content area on the web page consists of an `h1` heading. Let's change its color to orange, display its text content at 160 percent of the document's default size, and remove any top margin the browser may add to the heading by adding the following rule to the end of the style sheet.

```
/*Configure the size and color of level 1 headings*/
h1 {
  color: orange;
  font-size: 160%;
  margin-top: 0;
}
```

Next, let's begin configuring the display of the page's banner by adding the following rule to the end of the style sheet. The banner is made up of a dark gray heading with the website logo on it. In addition, a second graphic showing a computer and a stack of books will be displayed on it as well. You can use the `div` element whose `ID` is `logo` to reference the banner. Because this `div` element includes an embedded `img` element, that element will automatically be displayed on the resulting banner. Note that its position is set to relative. This will allow any content you place within it to be absolutely positioned.

Next, the height of the banner is specified and a small margin added to put a little extra space at the top of the banner and the top of the browser window. A little padding and a margin are then added to the `div` element, and a graphic file named background.png is used to repeatedly draw a graphic, which is 260 pixels tall by 10 pixels wide, repeatedly across the top of the browser window. Lastly, the `color` property is used to set the display of any character text to white.

Note that the background.png file used to draw the banner is only 260 pixels tall and 10 pixels wide. This is a relatively small file that can be downloaded quickly and repeatedly drawn over and over horizontally across the screen. This helps speed up the rendering of the page since small graphic files take less time to download.

```
/*Configure the location, height, and width of the document's logo*/
/*located at the left-hand side of the banner*/
#logo {
  position: relative;
  height: 260px;
  margin-top: 20px;
  padding: 10px;
  margin-bottom: 20px;
  background: url(background.png) repeat-x;
  color: white;
}
```

There are no plans to display on the banner. However, in the event that either of the graphic files that will be displayed on the banner becomes unavailable, each `img` element alternative text is displayed (in white instead of black).

With the presentation of the banner and the website's logo now configured, let's configure the display of the graphic image showing a computer and books on the banner by adding the following rule to the end of the style sheet. As you can see, the margin has been set to 0, positioning has been set to absolute, and the graphic's placement is specified using absolute positioning.

```
/*Specify the placement of the computer/book graphic at the top*/
/*of the screen*/
#concept {
  margin: 0;
  position: absolute;
  top: 50px;
  right: 40px;
  width: 30%;
  color: white;
  text-align: center;
}
```

Next, let's reserve space for the presentation of the page's content by adding the following rule to the end of the style sheet. Since the display of content is handled within a `div` element whose `ID` is `content`, this rule uses the `content ID` as its hook, allocating 70 percent of all available horizontal space to the display of content. This section of the web page is floated right.

```
/*Set the size of the area where content is displayed and float it*/
/*to the right*/
#content {
  width: 70%;
  float: right;
  padding-bottom: 20px;
  margin-left: 10px;
  padding: 2px;
}
```

The presentation of the web page's menu links is handled next by adding the following three rules to the end of the style sheet.

```
/*Set width of menu and float it to the left, disabling list style*/
/* and displaying a border on top of the first menu item*/
#menu {

  width: 25%;
  float: left;
  margin-top: 20px;
  margin-bottom: 20px;
  padding: 0;
  list-style: none;
  border-top: 2px solid gray;
}

/*Display a border below each menu item and increase menu font size*/
#menu li {
  border-bottom: 2px solid gray;
  font: 150% Arial, sans-serif;
  padding: 5px 0 5px 25px;
}

/*Disable the underlining of links*/
#menu a {
  text-decoration: none;
}
```

The first style rule allocates 25 percent of the available horizontal space to the `div` element that contains the menu's text links. This `div` element is floated to the left. Also, take note of the use of the `list-style` property, which is set to `none`. This hides the display of the menu list's bullets, which would otherwise be displayed. Also take note of the use of the `border-top` property, which has been configured to display a two-pixel thick solid gray line just above the first link.

By allocating 70 percent of available horizontal space to the content area and 25 percent for the display of the menu links, a total of 5 percent is left unaccounted for. Given that the menu links are floated to the left and the content is floated to the right, the leftover five percent represents unused white space in between these two entities.

The second rule shown above is responsible for displaying two-pixel thick solid gray lines under each link and for enlarging the display of link text. Lastly, a little extra padding is added to improve link presentation. The last rule shown above sets the `text-decoration` property to `none`. This eliminates the automatic underlining of links by the browser.

Last but not least, add the following style rule to the end of the style sheet. This rule begins by setting the `clear` property to `both`. This is needed because of the use of floated elements in the web document. Floated elements defy the normal flow of content presentation. When elements are floated, the browser is unable to determine the element's lower boundary. As a result, any content that follows can end up being displayed over the top of previously floated content. The use of the `clear` property prevents this from occurring by instructing the browser to set an arbitrary invisible line above which new content is not to be displayed.

```
/*Configure the display of the copyright statement*/
#copyright {
  clear: both;
  text-align: right;
  font-size: 90%;
  color: gray;
  padding-top: 20px;
  margin-bottom: 20px;
  background: url(separator.png) repeat-x;
}
```

After setting the value of the `clear` property, the rest of the rule sets the location, font size, and color of the copyright element and then uses the `background` property to display an orange graphic separator bar graphic along the bottom of the web page.

Okay, at this point the external style sheet for the website is complete. Figure 10.9 shows how the application of this style sheet has affected the presentation of the web page.

The presentation of content using the steps outlined above results in a fluid display that automatically adjusts its presentation based on the width of the browser window as demonstrated in Figure 10.10.

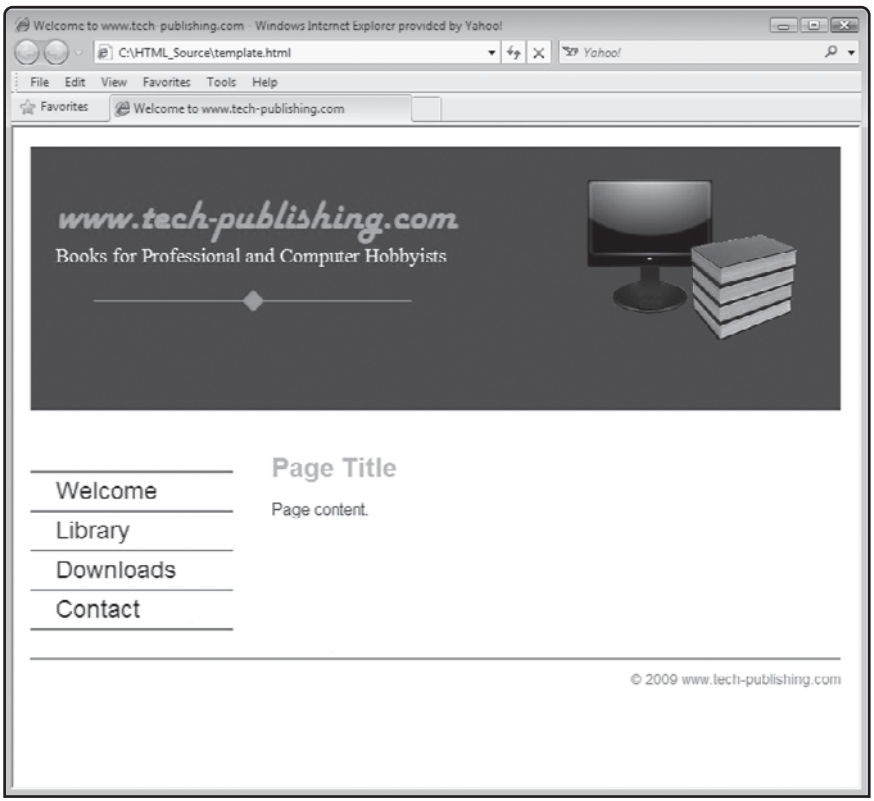

Figure 10.9

An example of how the website looks when viewed using a small browser window.

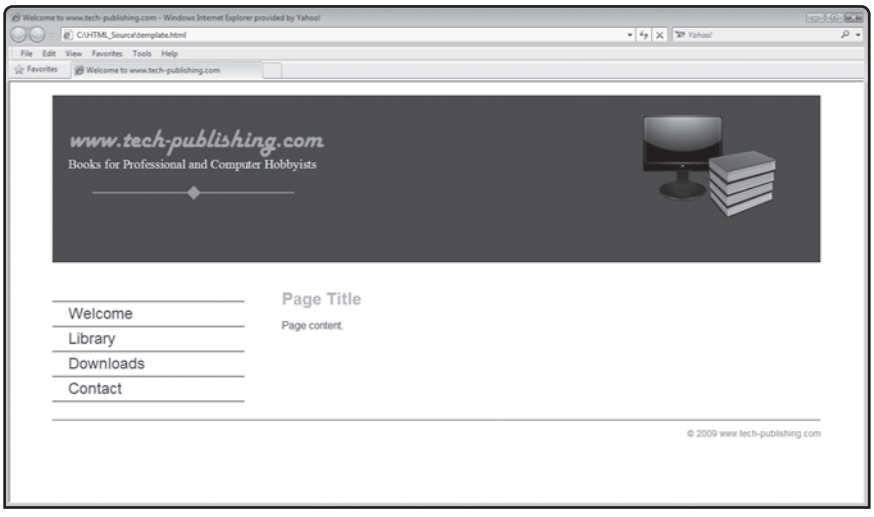

Figure 10.10

An example of how the website looks when viewed using a larger browser window.

Step 7: Assembling Document Files

At this point you have a template that you will use as the starting point for each of the website's four document files. In addition, you have an external style sheet that manages the presentation of any content defined in these pages. As you build out these web documents, you will find it necessary to return and add style rules to the external style sheet, specifically designed to handle the presentation of the new content that you are going to add. This is a normal and expected part of the development process.

Creating the index.html Document

The first web document to be created is the index.html document, which will be used to create the website's primary landing page. To create it, make a copy of the template.html page, renaming it index.html and then modify its markup by adding the elements and content highlighted below in bold exactly as shown.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

  <head>
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8" />
    <title>Welcome to www.tech-publishing.com</title>
    <link rel = "stylesheet" type = "text/css" href = "tech-pub.css" />
  </head>

  <body>

    <div id = "logo">
      <img src = "banner.png" width = "426" height = "192"
        alt = "Books for Professional and Computer Hobbyists" />
    </div>

    <div id = "concept">
      <img src = "pcandbooks.png" width = "210" height = "160"
        alt = "Picture of a computer and books" />
    </div>
```

```
    <ul id = "menu">
      <li><a href="index.html">Welcome</a></li>
      <li><a href="library.html">Library</a></li>
      <li><a href="downloads.html">Downloads</a></li>
      <li><a href="contact.html">Contact</a></li>
    </ul>

    <div id = "content">
      <h1>Welcome</h1>
      <p>Welcome to the grand opening of www.tech-publishing.com! I created
         this site to provide additional value and support to readers who
         have purchased or who are interested in purchasing one of my books.
         Here you will find information about my latest books and will
         be able to download the source code and graphic files for any of the
         programming projects covered in any books you have already
         purchased.
      </p>
      <p>
         Before you leave, please take the time to provide feedback on any
         books that you have purchased and share your thoughts and opinions on
         how any of this site's books can be improved. I would really
         like to hear what you have to say.
      </p>
      <p>Enjoy!</p>
      <p>Jerry Lee Ford, Jr.</p>
    </div>

    <div id = "copyright">
      &copy; 2009 www.tech-publishing.com
    </div>

  </body>

</html>
```

As you can see, all that has happened here is the modification of the document's `h1` element along with addition of a little content, embedded within paragraph elements.

Creating the library.html Document

Now let's move on to the creation of the library.html document. To create this document, make a copy of the template.html file and rename it library.html. Modify the document's markup by adding the elements and content highlighted below in bold print exactly as shown.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

  <head>
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8" />
    <title>Welcome to www.tech-publishing.com</title>
    <link rel = "stylesheet" type = "text/css" href = "tech-pub.css" />
  </head>

  <body>

    <div id = "logo">
      <img src = "banner.png" width = "426" height = "192"
        alt = "Books for Professional and Computer Hobbyists" />
    </div>

    <div id = "concept">
      <img src = "pcandbooks.png" width = "210" height = "160"
        alt = "Picture of a computer and books" />
    </div>

    <ul id = "menu">
      <li><a href="index.html">Welcome</a></li>
      <li><a href="library.html">Library</a></li>
      <li><a href="downloads.html">Downloads</a></li>
      <li><a href="contact.html">Contact</a></li>
    </ul>

    <div id = "content">
      <h1>Library</h1>
      <p>Scroll down to learn more about my latest books. If you find one
```

```
   you like, you can purchase it online by clicking on the link to
   amazon.com located at the bottom of this page.</p><br />
<p><img src = "xhtml-css.jpg" width = "93" height = "115"
   alt = "HTML, XHTML, and CSS for the Absolute Beginner" /></p>
<p><em>HTML, XHTML, &amp;amp; CSS For the Absolute Beginner</em> provides
   Beginner-level programmers with an entry-level introduction
   to HTML, XHTML, and CSS. This book serves as a primer for
   client-side web development. To make learning fun, the book
   uses a games-based instructional approach that provides working
   examples that demonstrate how HTML, XHTML, and CSS can be applied
   to the development of web pages and sites.</p><br />
<p><img src = "gamemaker.jpg" width = "93" height = "115"
   alt = "Getting Started with Game Maker" /></p>
<p><em>Getting Started with Game Maker</em> shows aspiring game
   developers how to create their very own, professional-quality computer
   games, no programming knowledge required. Using Game Maker and
   its drag-and-drop environment and following along with the
   step-by-step instructions, you will learn how to create
   arcade-style 2D and 3D games complete with graphics, sound
   effects, and music. Game Maker provides everything you need to
   create, test, debug, and run your games in a Windows
   environment.</p><br />
<p><img src = "ajax.jpg" width = "93" height = "115"
   alt = "Ajax Programming for the Absolute Beginner" /></p>
<p><em>Ajax Programming for the Absolute Beginner</em> teaches the
   principles of programming through simple game creation. You will
   acquire the skills that you need for more practical programming
   applications and learn how these skills can be put to use
   in real-world scenarios.</p><br />
<p><img src = "scratch.jpg" width = "93" height = "115"
   alt = "Scratch Programming for Teens" /></p>
<p><em>Scratch Programming for Teens</em> teaches you everything
   you need to know to get up and running quickly with Scratch.
   Scratch is a programming language intended to make programming
   easier to learn for novice programmers. It can be used to create
   computer games, interactive stories, graphic artwork, and
   computer animation, and all sorts of other multimedia projects.
   Scratch can also be used to play digital music and sound
```

```
        effects. If you aspire to one day become a professional
        programmer, Scratch provides everything you need to build a
        foundation.</p><br />
      <p><a href="http://www.amazon.com">Visit amazon.com to purchase a
        book</a></p>

    </div>

    <div id = "copyright">
      &copy; 2009 www.tech-publishing.com
    </div>

  </body>

</html>
```

As you can see, the content that has been added to the document consists primarily of paragraph and link elements as well as an `img` element for each of the four books that are displayed. The additional markup ends with the link to www.amazon.com at the bottom of the `div` element to which all of the new content has been added.

Creating the downloads.html Document

Next, let's create the downloads.html document. To create this document, make a copy of the template.html file and rename it downloads.html. Modify the document's markup by adding the elements and content highlighted below in bold print to it exactly as shown.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

  <head>
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8" />
    <title>Welcome to www.tech-publishing.com</title>
    <link rel = "stylesheet" type = "text/css" href = "tech-pub.css" />
  </head>

  <body>
```

```
<div id = "logo">
  <img src = "banner.png" width = "426" height = "192"
    alt = "Books for Professional and Computer Hobbyists" />
</div>

<div id = "concept">
  <img src = "pcandbooks.png" width = "210" height = "160"
    alt = "Picture of a computer and books" />
</div>

<ul id = "menu">
  <li><a href="index.html">Welcome</a></li>
  <li><a href="library.html">Library</a></li>
  <li><a href="downloads.html">Downloads</a></li>
  <li><a href="contact.html">Contact</a></li>
</ul>

<div id = "content">
  <h1>Downloads</h1>
  <p>To download all of the source code and graphic files needed to
     re-create the sample projects for a book you have already purchased,
     click on that book's link in the list below.</p>

  <ul id = "downloads">
    <li><a href="xhtml-css.zip">Download files for HTML, XHTML, and CSS for
         the Absolute Beginner</a></li>
    <li><a href="gamemaker.zip">Download file for Getting Started with
         Game Maker</a></li>
    <li><a href="ajax.zip">Download file for Ajax Programming for the
         Absolute Beginner</a></li>
    <li><a href="scratch.zip">Download file for Scratch Programming for
         Teens</a></li>
  </ul>

</div>

<div id = "copyright">
  &copy; 2009 www.tech-publishing.com
```

```
    </div>

  </body>

</html>
```

The additional markup that has been added to this document primarily consists of an unordered list containing a series of four links, each of which references a different zip file stored alongside the website's other content on the web server. When clicked, these links allow visitors to download and save a copy of the source code and graphic files for the author's books.

The addition of the new content in the downloads.html document requires that you go back and modify the tech-pub.css document by adding the style rules shown below to the end of the website's external style sheet.

```
/*Additional style rules for downloads.html*/

/*Set font size, type, add extra padding and set list style*/
#downloads li {
  font: 100% Arial, sans-serif;
  padding: 5px 0px 5px 0px;
  list-style: circle;
}

/*Disable the underlining of links*/
#downloads a {
  text-decoration: none;
}
```

As you can see, the first of these two new rules set and configure the size of the font used to display the download links, add a little padding around each list item (link), and then configure each link to display a circle bullet. The second rule removes the underlining of links.

Creating the contact.html Document

Now it is time to create the website's final document, contact.html. To create this document, make a copy of the template.html file and rename it contact.html. Modify the document's markup by adding the elements and content highlighted below in bold.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

  <head>
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8" />
    <title>Welcome to www.tech-publishing.com</title>
    <link rel = "stylesheet" type = "text/css" href = "tech-pub.css" />

    <script type = "text/javascript">
    <!-- Start hiding JavaScript statements
      function ProcessMsg() {
        if (document.getElementById("name").value != "") {
          if (document.getElementById("email").value != "") {
            if (document.getElementById("message").value != "") {
              document.getElementById("msgForm").submit();
            }
            else {
              window.alert("Error: You must enter a message.");
            }
          }
          else {
            window.alert("Error: You must enter your email address.");
          }
        }
        else {
          window.alert("Error: You must enter your name.");
        }
      }
    // End hiding JavaScript statements -->
    </script>

  </head>

  <body>

    <div id = "logo">
```

```
  <img src = "banner.png" width = "426" height = "192"
    alt = "Books for Professional and Computer Hobbyists" />
</div>

<div id = "concept">
  <img src = "pcandbooks.png" width = "210" height = "160"
    alt = "Picture of a computer and books" />
</div>

<ul id = "menu">
  <li><a href="index.html">Welcome</a></li>
  <li><a href="library.html">Library</a></li>
  <li><a href="downloads.html">Downloads</a></li>
  <li><a href="contact.html">Contact</a></li>
</ul>

<div id = "content">
  <h1>Contact</h1>
  <p>Have any thoughts on how this site or one of my books might be
     improved? I would like to hear them. Simply fill out the form
     below and click on Send.</p><br />
  <form id = "msgForm" method = "post" action = "process_form.php">
    <fieldset id = "submitmsg">
      <ol>
        <li><label for = "name">Name:</label>
            <input type = "text" id = "name" name = "name" />
        </li>
        <li><label for = "email">E-mail:</label>
            <input type = "text" id = "email" name = "email" />
        </li>
        <li><label for = "message">Message:</label>
            <textarea id = "message" name = "message" cols = "50"
              rows = "12"></textarea>
        </li>
        <li><label for = "submit"> Submit:</label>
          <input type = "submit" id = "submit" name = "submit"
            value = "Send" onclick = "ProcessMsg()"/>
        </li>
```

```
          </ol>
        </fieldset>
      </form>
    </div>

    <div id = "copyright">
      &copy; 2009 www.tech-publishing.com
    </div>

  </body>

</html>
```

As you can see, the additional markup added to this page is used to create a form that collects visitor feedback. Note that the opening `<form>` tag's `action` attribute points to a PHP file located on the web server (in the same folder as the html page). In addition, note that the `input` element configured as a Submit button includes an `onclick` event handler that executes a JavaScript function named `ProcessMsg()` when clicked.

The `ProcessMsg()` function uses the `getElementById()` method to retrieve any text typed into the form's `text` and `textarea` controls. If any of these controls are found to be empty, an error message is displayed, instructing the user what information has been omitted. If all required information has been provided, the `getElementById()` method is used to establish and reference the document's form and then to execute its `submit()` method.

The addition of the form and its various form elements requires that you go back and modify the tech-pub.css document one last time, adding the style rules shown below to the end of the website's external style sheet.

```
/*Additional style rules for contact.html*/

/*Hide the fieldset element's border*/
#submitmsg {
  border: none;
}

/*Disable the display of list numbering*/
#submitmsg ol {
  list-style: none;
  margin: 0;
  padding: 0;
```

```
}

/*Add some space between form fields*/
#submitmsg li {
  margin-bottom: 10px;
}

/*Set width of labels to 15 percent and configure them*/
#submitmsg label {
  width: 15%;
  float: left;
  text-align: right;
  margin-right: 5%;
}

/*Set the width of the form button*/
#submit {
  min-width: 80px;
}
```

The first of these five style rules removes the display of the default border that is normally displayed by the `fieldset` element. The form's elements are organized and displayed as an ordered list. The second rule disables the display of the list's numbers and eliminates any margin or padding that the browser may add to the list. The third rule adds a little space between form controls by adding a padding of 10 pixels to the end of each list element. The fourth rule modifies the presentation of form labels, configuring them so that label text is left aligned. By assigning each label the same width, the controls also end up aligned when displayed. Finally, the fifth and final rule sets the minimum width assigned to the form's Submit button.

In order for the contact.html document to work as previously explained, you must have a program named process_form.php located in the same folder on your web server as your web documents. The process_form.php program is a PHP script. PHP is a server-side scripting language used in the creation of web applications. A discussion of PHP is well outside the scope of this book.

PHP is one of the most universally supported server-side scripting languages. Chances are that your web host supports it. Assuming this is the case, all you have to do to enable the proper operation of the Contact form is to create a text file named process_form.php and then add the following statement to it.

While I will not go into any detail describing what this PHP script does, I will point out that it consists of two parts. The first part is written in PHP and its purpose is to process data submitted to it by the contact.html page's form. The second part of the script contains a copy of the template.html page. Its content has been modified to display a message that lets visitors know that their messages have been submitted. This XHTML content is returned by the PHP program to the browser, where it is displayed in the browser window.

```
<?
if (($_POST[name] == "") ||
    ($_POST[email] == "") ||
    ($_POST[message] == "")) {
    header("Location: contact.html");
    exit;
}

$msg = "Email sent  from www.tech-publishing.com\n";
$msg .= "Sender Name:\t$_POST[name]\n";
$msg .= "Sender Email:\t$_POST[email]\n";
$msg .= "Message:\t$_POST[message]\n";

$to = "jlf04@yahoo.com";
$subject = "Reader Feedback";
$mailheaders = "From: www.tech-publishing.com\n";
$mailheaders .= "Reply-To: $_POST[sender_email]\n";
mail($to, $subject, $msg, $mailheaders);
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

  <head>
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8" />
    <title>Welcome to www.tech-publishing.com</title>
    <link rel = "stylesheet" type = "text/css" href = "tech-pub.css" />
  </head>
```

```
  <body>

    <div id = "logo">
      <img src = "banner.png" width = "426" height = "192"
        alt = "Books for Professional and Computer Hobbyists" />
    </div>

    <div id = "concept">
      <img src = "pcandbooks.png" width = "210" height = "160"
        alt = "Picture of a computer and books" />
    </div>

    <ul id = "menu">
      <li><a href="index.html">Welcome</a></li>
      <li><a href="library.html">Library</a></li>
      <li><a href="downloads.html">Downloads</a></li>
      <li><a href="contact.html">Contact</a></li>
    </ul>

    <div id = "content">
      <h1>Contact</h1>
      <p>Your message has been sent. Thanks!</p><br />
    </div>

    <div id = "copyright">
      &copy; 2009 www.tech-publishing.com
    </div>

  </body>

</html>
```

Step 8: Testing the New Website

Well, that's it! As long as you followed along carefully with the instructions that have been provided, your copy of the www.tech-publishing.com website should be ready for testing. Start by loading the index.html document into your web browser and then explore the site and make sure that everything works as advertised.

A complete copy of the source code for this project, including its style sheet, and the graphics needed to create its graphic controls is available on the book's companion web page, located at www.courseptr.com/downloads.

Summary

Congratulations on making it through to the end of this book! By making it this far, you have learned the fundamentals of web development using HTML, XHTML, and CSS. Not only that but you have a basic understanding of the document object model (DOM) and a good understanding of JavaScript programming. By mastering the basics of all of these web development technologies, you have positioned yourself to make the leap from web surfer to web developer. As a result, you are ready to begin developing your own websites and to begin making a name for yourself by carving out your own little piece of space on the World Wide Web.

Before you put this book down and move on to other things, why don't you take a few minutes to implement the following challenges?

Challenges

1. **Using the www.tech-publishing.com website as a starting point, create a new website of your own.**
2. **Replace the graphics and modify the color scheme and layout to suit your own preferences.**
3. **If you don't already have one, find a website host and upload your new website for the world to see.**