CHAPTER 9

# Site Navigation

When you complete this chapter, you will be able to:

- Create usable navigation
- Build text-based navigation
- Use graphics for navigation and linking
- Use lists for navigation
- Build horizontal navigation bars
- Build vertical navigation bars
- Using background color and graphics to enhance navigation
- Create hover rollovers

The free-flowing nature of information in a nonlinear hypertext environment can be confusing to navigate. Help your users find content easily rather than making them hunt through a maze of choices. Let your users know where they are at all times and where they can go within your Web site. In this chapter, you learn to build user-focused navigation to accomplish these goals.

## Creating Usable Navigation

Webopedia (*www.webopedia.com*) defines hypertext as "a system in which objects (text, pictures, music, programs, and so on) can be creatively linked to each other…. You can move from one object to another even though they might have very different forms." Hypertext was envisioned in the 1960s by Ted Nelson, who described it as nonsequential writing in his book *Literary Machines*. Nelson's basic idea of connecting content through hypertext linking influenced the creators of the Web. With hypertext-linked content, users can traverse information in any order or method they choose, creating their own unique views.

Hypertext is a distinctly different environment in which to write and structure information. In traditional paper-based media, users navigate by turning pages or referring to a table of contents or an index separate from the information they are reading. In a hypertext document, users can connect instantly to related information. The hypertext forms of traditional navigation devices, such as tables of contents and cross-references, can be displayed constantly alongside related content. The user can explore at will, jumping from one point of interest to another. Of course, the ease of navigation depends on the number of links and the context in which they were added by the hypertext author.

In HTML, hyperlinks are easy to create and add no extra download time when they are text based. When you are planning your site navigation, do not skimp on navigation cues, options, and contextual links. You can use the CSS style properties you have learned about to create attractive navigation elements. If you use graphics for navigation, remember to keep your navigation graphics simple and reuse the same graphics throughout your Web site. Once the navigation graphics are loaded in the user's cache, the server does not have to download them again. Use an alternate set of text links in the event that the user cannot or will not view your graphics and to meet accessibility guidelines. You will learn more about text links later in this chapter.

Effective navigation includes providing not only links to other pages in the Web site, but also cues to the user's location. Users should be able to answer the following navigation questions:

- Where am I?
- Where can I go?
- How do I get there?
- How do I get back to where I started?

To allow users to answer these questions, provide the following information:

- The current page and the type of content they are viewing
- Where they are in relation to the rest of the Web site
- Consistent, easy-to-understand links
- Alternatives to the browser's Back button that let users return to their starting point

## Locating the User

Figure 9-1 shows a page from the National Archives Web site (*www.archives.gov*) that displays a number of user-orienting features.


**Figure 9-1** Providing user location cues

The navigation cues on this page offer many options without disorienting the user. A search option lets the user search the entire site. A linked **breadcrumb path** at the top of the page shows the user's location within the site hierarchy. Users can click any of the links in the path to move through the content structure. This location device is especially effective in guiding users who may have arrived at this page from somewhere outside this Web site. The section heading identifies the current section, and the links on the left let the user jump to related content pages. Using these navigation devices, users can choose to jump directly to a page, search for information, or move back up through the information hierarchy.

## Limiting Information Overload

Many Web sites tend to present too much information at one time. Lengthy files that require scrolling or have arrays of links and buttons can frustrate and overwhelm the user. You can limit information overload in the following ways:

- *Create manageable information segments*—Break your content into smaller files, and then link them together. Provide logical groupings of choices. Keep a flat hierarchy. A good rule to follow is that users should not have to click more than two or three times to get to the information they desire.
- *Control page length*—Do not make users scroll through never-ending pages. Long files also can mean long downloads. Provide plenty of internal links to help users get around, and keep the pages short. You can judge your page length by pressing the Page Down key; if you have to press it more than two or three times to move from the top to the bottom of your page, break up the content.
- *Use hypertext to connect facts, relationships, and concepts*—Provide contextual linking to related concepts, facts, or definitions, letting the users make the choices they want. Know your material, and try to anticipate the user's information needs.

# Activity: Building Navigation Structures

Text-based linking often is the most effective and accessible way to provide navigation on your site. In the following set of steps, you will link a series of sample Web pages using text-based navigation techniques. Figure 9-2 shows the structure of the collection of sample HTML documents that you will use, including the Home page, Table of Contents page, Site Map page, and individual Chapter pages.

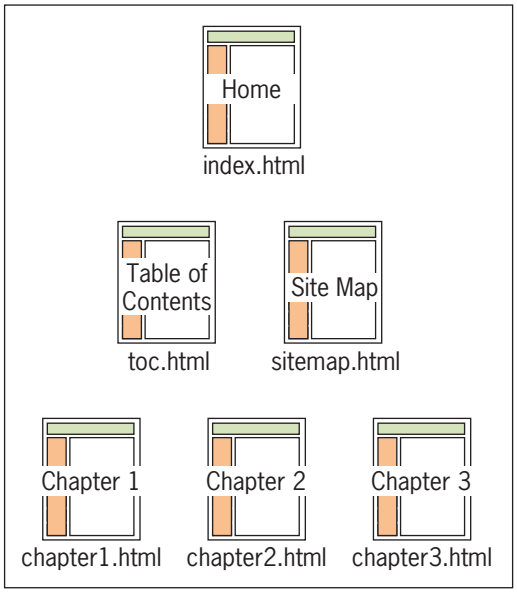


**Figure 9-2** Sample file structure

This book's Online Companion Web site contains all the HTML files for the sample Web site illustrated in Figure 9-2.

In the hypertext environment, the user should be able to select links in the table of contents to jump to any document in the collection. In the following steps, you will add a variety of linking options that produce different paths through the information. The focus for these steps is the Table of Contents page, toc.html, and how it relates to the rest of the content in the collection. You will also add navigation options to the individual chapter pages. The Index and Site Map pages are included to complete the sample Web site and will be target destinations for some of the links you will build. You can then apply these linking techniques to your own Web site projects.

To complete the steps in this chapter, you need to work on a computer with a browser and an HTML editor or a simple text editor, such as Notepad or SimpleText.

To prepare for linking the Web pages:

1. Copy the following files from the Chapter09 folder provided with your Data Files:
   - index.html
   - toc.html
   - sitemap.html
   - chapter1.html

- chapter2.html
- chapter3.html
- styles.css

2. Save the files in the Chapter09 folder in your work folder using the same filenames. (Create the Chapter09 folder, if necessary.) Make sure you save all the files in the same folder.

In the following set of steps, you use common HTML coding techniques. If necessary, review your basic HTML knowledge before proceeding with these instructions.

## Linking with a Text Navigation Bar

The Table of Contents page must link to the other main pages of the Web site, allowing users to go directly to the pages they want. You can achieve this by adding a simple text-based navigation bar.

To build the navigation bar:

1. From the Chapter09 folder in your work folder, open the file **toc.html** in your browser. It should look like Figure 9-3.

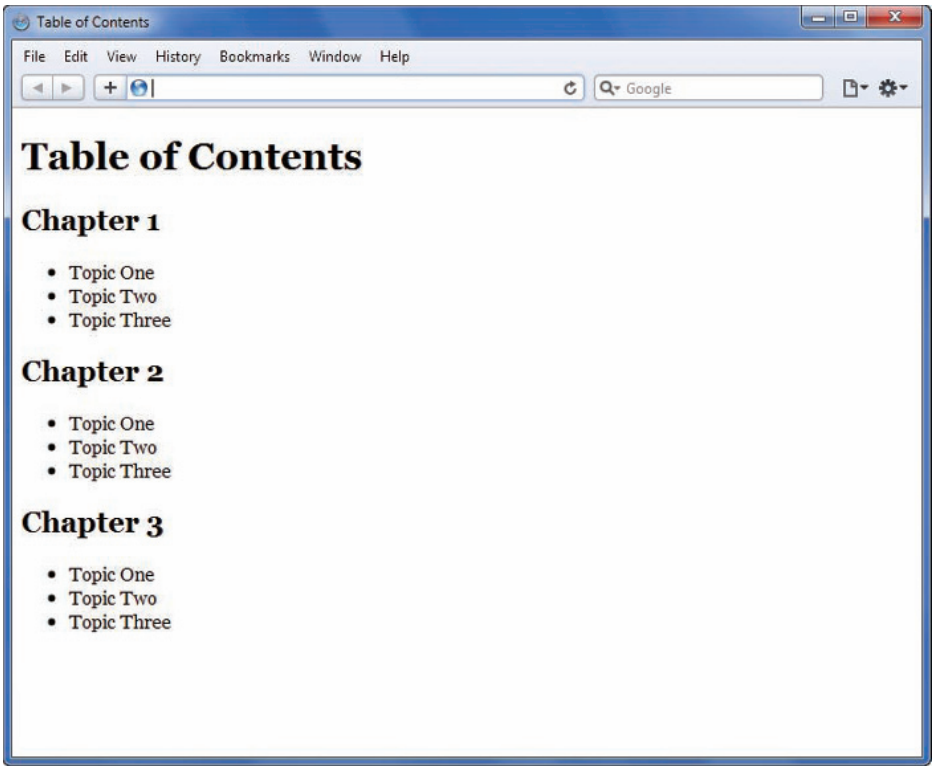


**Figure 9-3** Original HTML file

2. Open the file in your HTML editor, and examine the code. Notice that the file contains a link to an external style sheet that controls the basic styles for the site. The complete code for the page follows:

```
<!DOCTYPE html>
<html>
<head>
<title>Table of Contents</title>
<meta content="text/html; charset=utf-8" http-equiv=
   "Content-Type" />
<link href="styles.css" rel="stylesheet" type="text/css" />

</head>
<body>
<h1>Table of Contents</h1>

<h2>Chapter 1</h2>
   <ul>
      <li>Topic One</li>
      <li>Topic Two</li>
      <li>Topic Three</li>
   </ul>
<h2>Chapter 2</h2>
   <ul>
      <li>Topic One</li>
      <li>Topic Two</li>
      <li>Topic Three</li>
   </ul>
<h2>Chapter 3</h2>
   <ul>
      <li>Topic One</li>
      <li>Topic Two</li>
      <li>Topic Three</li>
   </ul>
</body>
</html>
```

In this set of steps, you use the id selector to provide an identifying name for the  element. You can use the id name to apply CSS styles to a section of a Web page. Refer to Chapter 4 to read more about the id selector.

3. Add a <p> element to place the navigation bar immediately following the opening <body> tag. Set the id attribute to *headernav* as shown in the following code. (The new code you should add appears in blue in this step and the following steps.)

```
<body>
<p id="headernav"> </p>
```

4. Add text within the new <p> element as shown in the following code.

```
<p id="headernav">Home | Table of Contents | Site Map</p>
```

5. Add <a> tags with href attributes that link to the home page and the site map.

```
<p id="headernav">
<a href="index.html">Home</a> | Table of Contents |
<a href="sitemap.html">Site Map</a></p>
```

6. Add a span element with a class attribute to contain the Table of Contents text. Because this is the Table of Contents page, the text "Table of Contents" is not a hypertext link but is bold to designate the user's location. The code looks like this:

```
<p id="headernav">
<a href="index.html">Home</a> |
<span class="current">Table of Contents</span> |
<a href="sitemap.html">Site Map</a></p>
```

7. Open the stylesheet file **styles.css** in your editor. When you open the file it looks like this:

```
/* Principles of Web Design 5th ed. */
/* style sheet for linking activity in Chapter 9 */


body {font-family: georgia, serif;}
li {line-height: 125%;}
p {margin: 0px 20px 0px 30px}
```

8. In styles.css, add a style for the id *headernav* that specifies a width, automatic margins, padding, a border, and text alignment. Use a comment to describe the style rule.

```
/* Navigation Header */
#headernav {
   margin-left: auto;
   margin-right: auto;
   padding: 10px;
   border: solid 1px black;
   width: 300px;
   text-align: center;
}
```

9. Add another style for the *current* class that indicates which page the user is currently viewing.

```
/* Current Page Indicator */
.current {
   font-weight: bold;
}
```

10. Save and close the style sheet file, and then view the Table of Contents page in your browser. It should look like Figure 9-4. Test your hypertext links in the navigation bar to make sure they point to the correct pages.

11. Add this navigation bar to all of the sample pages. Remember to change the links and placement of the  element to reflect the current page. For example, the Site Map page would have the text "Site Map" contained in the  and links to the Table of Contents and Home pages. The chapter pages (chapter1.html, chapter2.html, and chapter3.html) do not need a span element, just links to the Home, Table of Contents, and Site Map pages.

12. Save **toc.html** in the Chapter09 folder of your work folder, and leave it open in your HTML editor for the next steps.

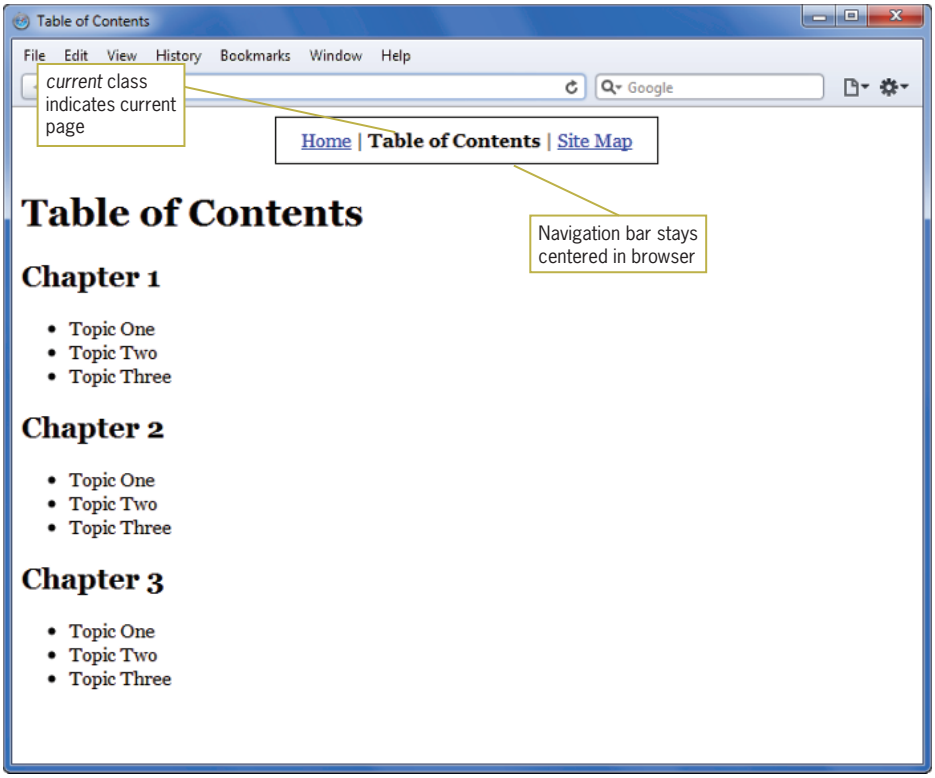


**Figure 9-4** Adding a text-based navigation bar

## Linking to Chapter Pages

While the navigation bar lets users access the main pages in the Web site, the table of contents lets users access the exact content pages they want. The Table of Contents page therefore needs links to the individual chapter files in the Web site. In this set of steps,

you will add links to the individual chapter files listed in the table of contents.

To build page links:

1. Continue working in the file **toc.html**. Add the following <a> element around the text "Chapter 1":

```
<h2><a href="chapter1.html">Chapter 1</a></h2>
```

2. Add similar <a> elements around the text "Chapter 2" and "Chapter 3" that point to the files chapter2.html and chapter3.html, respectively.

3. Save **toc.html** and view the finished Table of Contents page in your browser. It should look like Figure 9-5. Test your hypertext links to make sure they point to the correct page.

As this example shows, remember always to make <a> the innermost set of tags to avoid extra space in the hypertext link.

This linking method lets the users scroll through the table of contents to scan the chapters and topics and then jump to the chapter they want. The link colors—by default, blue for new and purple for visited—allow users to keep track of which chapters they already have visited.

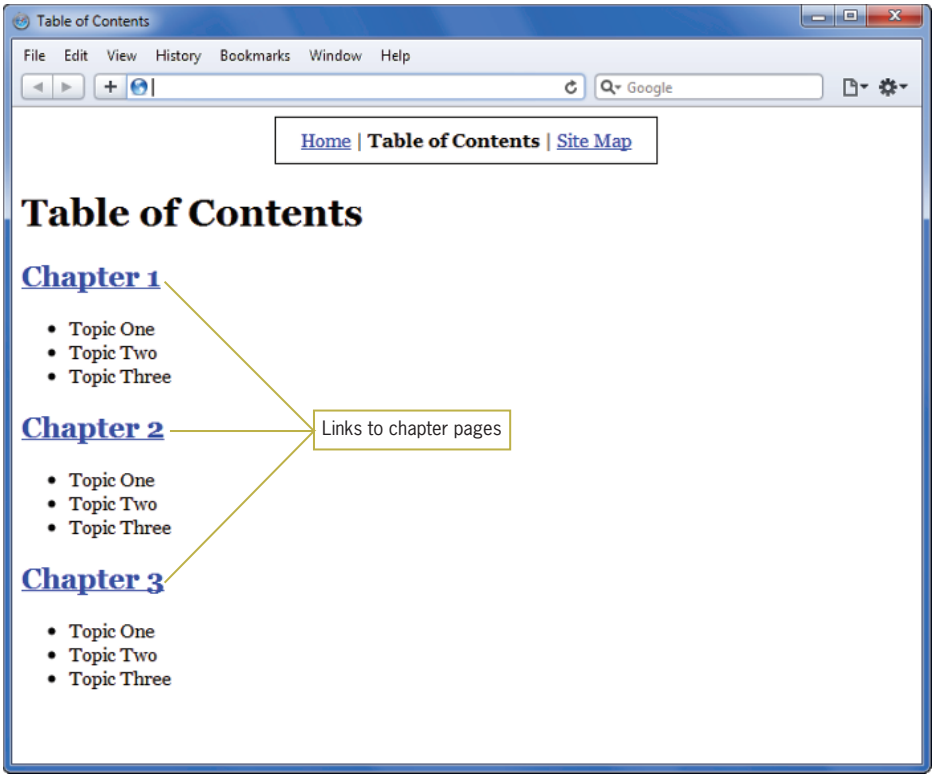


**Figure 9-5** Adding chapter links

## Adding Internal Linking

In addition to linking to external documents, you also can add internal links for navigating within the table of contents itself. In the Table of Contents page illustrated in Figure 9-5, you will add a Back to Top link that lets users return to the top of the page when they reach the bottom of the page.

This requires two <a> anchor elements: one uses the name attribute to name a **fragment identifier** in the document; the other targets the fragment name in the href attribute.

To add an internal link:

1. Continue working with the file **toc.html** in your editor. Add a new <a> element at the top of the page, immediately after the <body> tag. Add a name attribute, and set the value to *top* as shown.

```
<body>
<a name="top"></a>
<p id="headernav">
<a href="index.html">Home</a> |
<span class="current">Table of Contents</span> |
<a href="sitemap.html">Site Map</a></p>
```

   Notice that this <a> element is empty. The name attribute identifies this location in the document as "top." You can then refer to this name as an href target elsewhere in the document. The value of the name attribute can be any combination of alphanumeric characters that you choose.

2. Add an <a> element at the bottom of the page, after the listing for Chapter 3 and just before the closing </body> tag. Reference the target fragment *top* by using the number sign ( # ) in the href attribute, as shown in the following code.

```
<h2><a href="chapter3.html">Chapter 3</a></h2>
   <ul>
    <li>Topic One</li>
    <li>Topic Two</li>
    <li>Topic Three</li>
   /ul>
<a href="#top">Back to Top</a>
</body>
```

3. Add a <p> element around the <a> element, and set the id attribute to *footer* to identify the division. This will come in handy if you decide to add a style to the footer later.

```
<p id="footer"><a href="#top">Back to Top</a></p>
```

4. Save the **toc.html** file, and view the Table of Contents page in your browser. Resize your browser to show only a portion of the page, as shown in Figure 9-6.

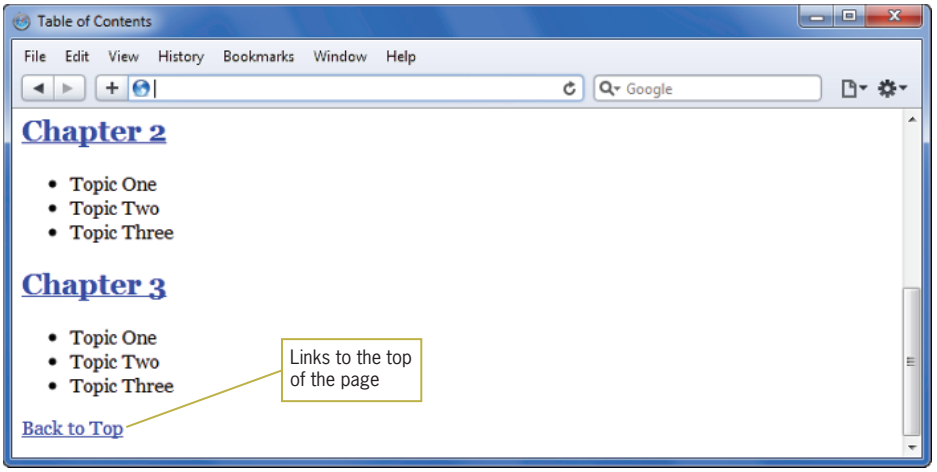


**Figure 9-6** Adding a Back to Top link

5. Test the link to make sure it opens the browser window at the top of the page, as shown in Figure 9-7.

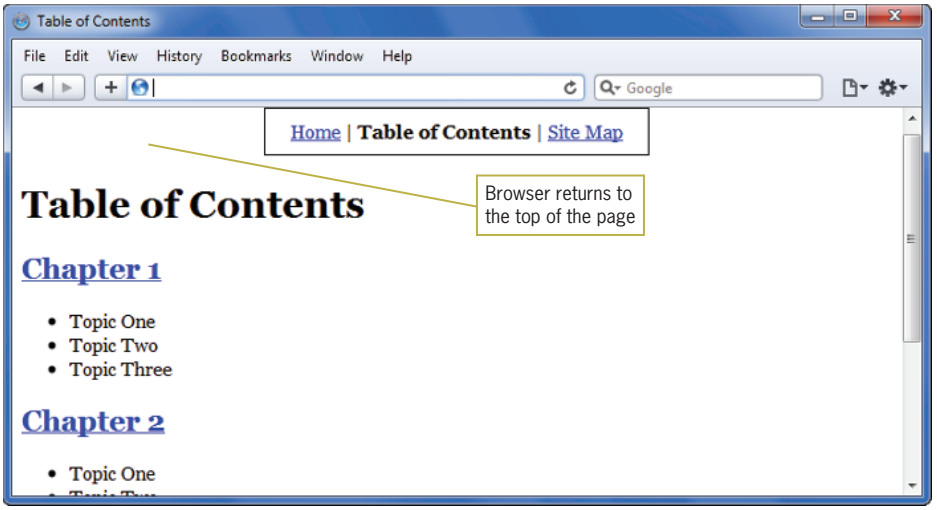


**Figure 9-7** Results of testing the Back to Top link

## Adding a Page Navigation Bar

You can use additional fragment identifiers in the table of contents to add more user-focused navigation choices. Figure 9-8 shows the addition of a page navigation bar.

When users click one of the linked chapter numbers, they jump to the specific chapter information they want to view within the table of contents further down the page.

To add a page navigation bar:

1. Continue working with the file **toc.html** in your HTML editor. Add a <p> element immediately after the <h1> element as shown. Set the id attribute to *pagenav* as shown in the following code.

```
<h1>Table of Contents</h1>
<p id="pagenav"> </p>
```

2. Add text to the <p> element as shown in the following code.

```
<p id="pagenav">
Jump down this page to Chapter... 1 | 2 | 3</p>
```

3. Add <a> elements around each chapter number within the <p>. Insert href attributes that point to a named fragment for each chapter.

```
<p id="pagenav" style="text-align: center;">
Jump down this page to Chapter...
<a href="#chapter1">1</a> |
<a href="#chapter2">2</a> |
<a href="#chapter3">3</a></p>
```

4. Add the name attribute to each chapter's <a> element. These are the fragment names you referred to in Step 3. The following code shows the new name attributes in the <a> element for each chapter.

```
<h2><a href="chapter1.html" name="chapter1">Chapter 1</a>
</h2>
   <ul>
      <li>Topic One</li>
      <li>Topic Two</li>
      <li>Topic Three</li>
   </ul>
<h2><a href="chapter2.html" name="chapter2">Chapter 2</a>
</h2>
   <ul>
      <li>Topic One</li>
      <li>Topic Two</li>
      <li>Topic Three</li>
   </ul>
<h2><a href="chapter3.html" name="chapter3">Chapter 3</a>
</h2>
   <ul>
      <li>Topic One</li>
      <li>Topic Two</li>
      <li>Topic Three></li>
   </ul>
```

5. Save the **toc.html** file in the Chapter09 folder in your work folder, and then close the file. View the Table of Contents page in your browser. Resize your browser to show only a portion of the page, as shown in Figure 9-8.

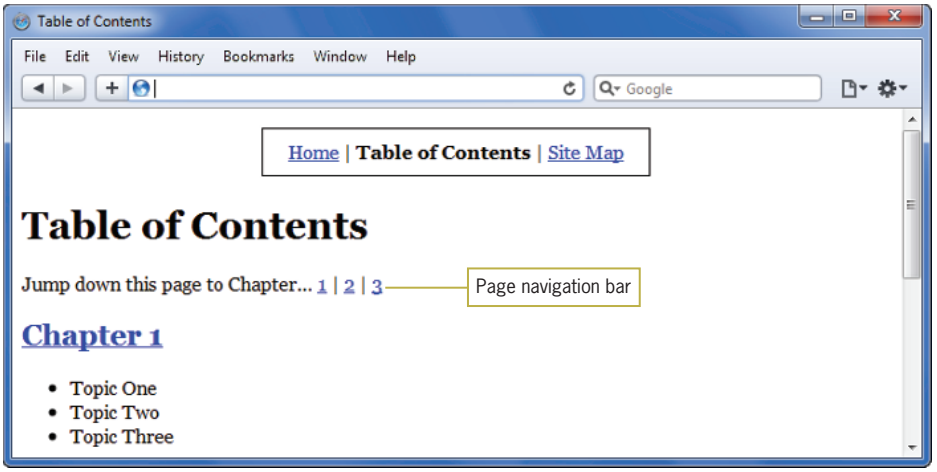


**Figure 9-8** Adding a page navigation bar

6. Test the navigation bar links by selecting a chapter number and making sure the browser window opens to the correct place in the file. Figure 9-9 shows the result of selecting the Chapter 2 link.

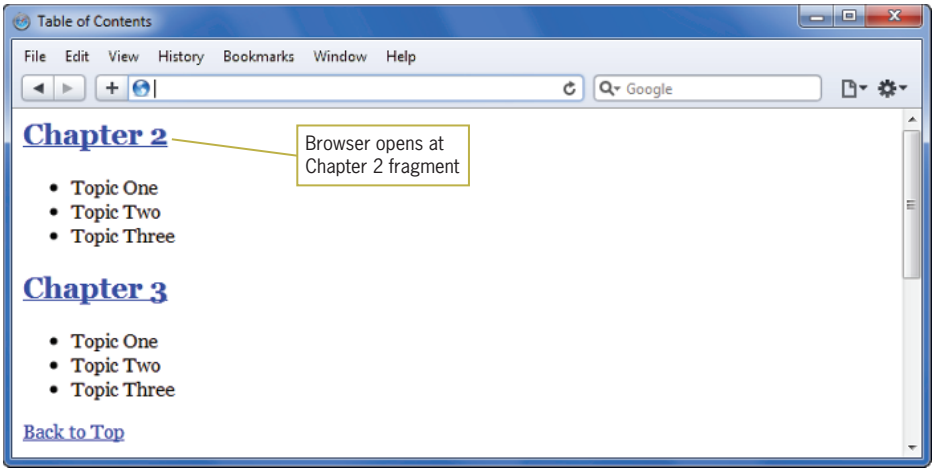


**Figure 9-9** Results of testing the page Chapter 2 link

## Linking to External Document Fragments

Now that you have completed internal linking in the table of contents, reexamine how the table of contents is linked to each chapter file. Currently, each chapter has one link in the table of contents; users click the chapter link, and the browser opens the chapter file at the top. However, each chapter also contains multiple topics, which can be linked as external fragments. You can let users jump from the table of contents to the exact topic they want within each chapter. This requires adding code to both the Table of Contents page and each individual chapter page.

To add links to external fragments:

1. In your HTML editor, open the file **chapter1.html** from the Chapter09 folder in your work folder.
2. Find the topic headings within the file. For example, the following code creates the topic heading for Topic 1:

   ```
   <h2>Topic 1</h2>
   ```

3. Add an <a> element around the text "Topic1." Set the name attribute value to *topic1* for this heading, as shown in the following code.

   ```
   <h2><a name="topic1">Topic 1</a></h2>
   ```

4. Now add the same code to each of the other topic headings in the file, using *topic2* and *topic3* as the name attribute values for the Topic 2 and Topic 3 headings, respectively.
5. Save **chapter1.html** in the Chapter09 folder of your work folder, and then close the file.
6. In your HTML editor, open the files **chapter2.html** and **chapter3.html** from the Chapter09 folder of your work folder. Repeat Steps 2 through 5 for both files, adding appropriate name attribute values for each topic in each file.
7. In your HTML editor, open the file **toc.html** from the Chapter09 folder of your work folder. (This is the toc.html file you saved in the last set of steps.)

8. Find the topic links for each chapter within the file. For example, the following code produces three topic links for Chapter 1:

```
<h2>
<a href="chapter1.html" name="chapter1">Chapter 1</a>
</h2>
   <ul>
      <li>Topic One</li>
      <li>Topic Two</li>
      <li>Topic Three</li>
   </ul>
```

9. Add <a> elements for each topic. The href value is the filename combined with the fragment name. The following code shows the <a> element for the Chapter 1, Topic 1 link.

```
<h2>
<a href="chapter1.html" name="chapter1">Chapter 1</a>
</h2>
   <ul>
      <li><a href="chapter1.html#topic1">Topic One</a></li>
      <li>Topic Two</li>
      <li>Topic Three</li>
</ul>
```

10. Continue to add similar links for each topic listed in the table of contents. When you are finished, your chapter navigation section should look like the following example. The link code is shown in color. (The location of your line breaks might differ.)

```
<h2>
<a href="chapter1.html" name="chapter1">Chapter 1</a>
</h2>
   <ul>
      <li><a href="chapter1.html#topic1">Topic One</a></li>
      <li><a href="chapter1.html#topic2">Topic Two</a></li>
      <li><a href="chapter1.html#topic3">Topic Three</a>
         </li>
   </ul>
<h2>
<a href="chapter2.html" name="chapter2">Chapter 2</a>
</h2>
   <ul>
      <li><a href="chapter2.html#topic1">Topic One</a></li>
      <li><a href="chapter2.html#topic2">Topic Two</a></li>
      <li><a href="chapter2.html#topic3">Topic Three</a>
         </li>
   </ul>
<h2>
<a href="chapter3.html" name="chapter3">Chapter 3</a>
</h2>
   <ul>
      <li><a href="chapter3.html#topic1">Topic One</a></li>
```

```
        <li><a href="chapter3.html#topic2">Topic Two</a></li>
        <li><a href="chapter3.html#topic3">Topic Three</a>
        </li>
    </ul>
```

11. Save the **toc.html** file in the Chapter09 folder of your work folder, and view it in your browser. Test the topic links by selecting a chapter topic and making sure the browser window opens to the correct place in the correct file. Figure 9-10 shows the result of selecting the Chapter 2, Topic 2 link.

    When users click the topic links in the table of contents, the browser opens the destination file and displays the fragment.

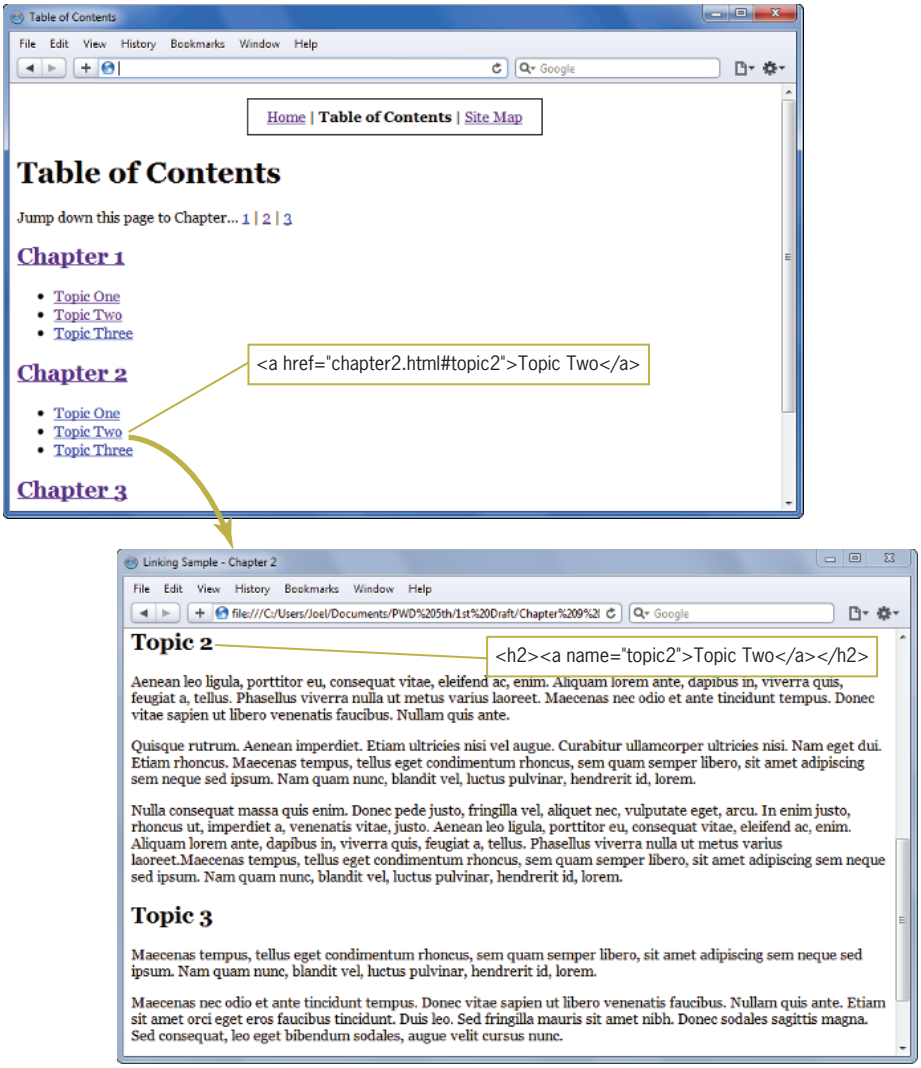


**Figure 9-10** Linking to an external fragment

## Adding Page Turners

Each chapter file currently contains a navigation bar and fragment identifiers for each topic within the chapter. In this page collection, the user can jump to any file and topic within a file, though some users may want to read the pages sequentially. You can offer this function by adding page-turner links. Page turners let you move either to the previous or next page or section in the collection. These work well in a linear structure of pages, for computer-based learning, or where users read pages or sections in order as shown in Figure 9-11.

Note that Chapter 1 uses the table of contents as the previous page, while Chapter 3 uses the site map as the next page.

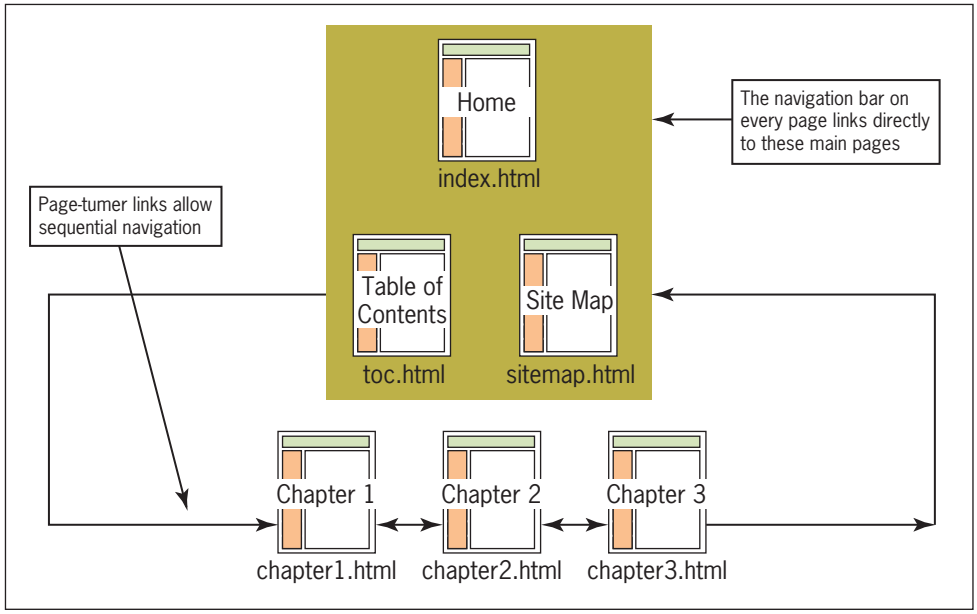


**Figure 9-11** Sequential page turning

To add page-turner links:

1. In your editor, open the file **chapter1.html** from the Chapter09 folder of your work folder.
2. Add a <p> element to place the page turner links at the bottom of the page. Set the id attribute to *footernav* as shown in the following code. Insert the code immediately before the closing </body> tag.

```
<p id="footernav">Previous | Chapter 1 | Next</p>
</body>
```

3. Add an <a> element for the previous page. Because this is Chapter 1, make the previous page destination toc.html, as shown in the following code.

```
<p id="footernav"><a href="toc.html">Previous</a> |
   Chapter 1 | Next</p>
```

4. Add an <a> element for the next page. Because this is Chapter 1, make the next page destination chapter2.html, as shown in the following code.

```
<p id="footernav"><a href="toc.html">Previous</a> |
   Chapter 1 |
<a href="chapter2.html">Next</a></p>
```

5. Add a span element with a class attribute to contain the Chapter 1 text. Because this is the Chapter 1 page, the text "Chapter 1" is not a hypertext link but is bold to designate the user's location. You will reuse the *current* style you used previously when building the header navigation bar to indicate the current page. The code looks like this:

```
<p id="footernav"><a href="toc.html">Previous</a> |
<span class="current">Chapter 1</span> |
<a href="chapter2.html">Next</a></p>
```

6. Save the **chapter1.html** file in the Chapter09 folder of your work folder.

7. Now that the HTML code is complete, you will need to add the *footernav* style to the CSS style sheet. Open the stylesheet file **styles.css** in your editor. Locate the headernav style:

```
/* Navigation Header */
#headernav {
   margin-left: auto;
   margin-right: auto;
   padding: 10px;
   border: solid 1px black;
   width: 300px;
   text-align: center;
}
```

8. You can easily match the style of the header navigation bar for the footer by simply adding the *footernav* id selector to the existing style rule as shown:

```
/* Navigation Header */
#headernav, #footernav {
   margin-left: auto;
   margin-right: auto;
   padding: 10px;
   border: solid 1px black;
   width: 300px;
   text-align: center;
}
```

9. Save and close the **styles.css** file. Save the **chapter1.html** file, and then view it in your browser. Your file should now look like Figure 9-12. Test the page-turner links to make sure they point to the correct files.

10. Add the page-turner links to chapter2.html and chapter3.html, changing the Previous and Next links to point to the correct files. Test all your links to make sure they work properly. When you are finished, save and close all the .html files in the Chapter09 folder of your work folder.

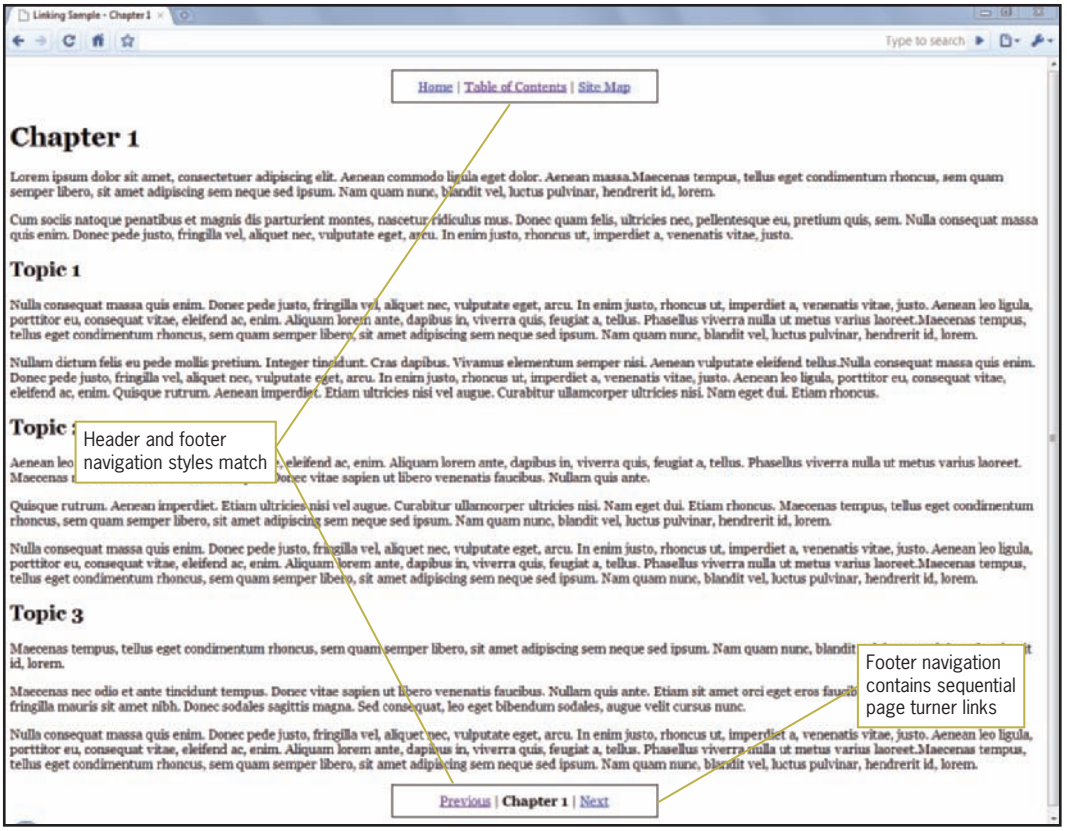


**Figure 9-12** Adding page turners in the navigation bar

## Adding Contextual Linking

One of the most powerful hypertext capabilities is contextual linking. **Contextual links** allow users to jump to related ideas or cross-references by clicking the word or item that interests them. These are standard hypertext links that you can embed directly in the flow of your content by choosing the key terms and concepts

you anticipate your users will want to follow. Figure 9-13 shows a page from the Wikipedia Web site (*www.wikipedia.com*) that contains contextual linking.

Note the links within the lines of text, which let the user view related information in context. Including the link within a line of text is more effective than including a list of keywords, because users can see related information within the context of the sentence they are reading. Users also can see that repeated words are linked no matter how many times they appear within the browser window, offering users the opportunity to access additional information at any time.


**Figure 9-13** Contextual linking

## Navigation Summary

You can choose from a variety of navigation options to link a collection of pages. The sample Web pages in this section demonstrate the following text-based linking actions:

- To main pages (Home, Table of Contents, Index)
- To the top of each chapter

- Within the Table of Contents page to chapter descriptions
- From the Table of Contents page to specific topics within each chapter
- Between previous and next chapters
- To related information by using contextual links

Use as many of these options as necessary, but remember to view your content from the user's perspective. Use enough navigation options to allow easy and clear access to your content.

## Using Graphics for Navigation and Linking

Although the current Web design trends are towards text-based navigation, there are still many instances where creating a clickable graphic for a link is desirable. To make sure your navigation graphics help rather than hinder your users, use the same graphics consistently throughout your Web site, for the following reasons:

- *To provide predictable navigation cues for the user*—After users learn where to find navigation icons and how to use them, they expect them on every page. Consistent placement and design also build users' trust and help them feel confident that they can find the information they want.
- *To minimize download time*—After the graphic is downloaded, the browser retrieves it from the cache for subsequent pages rather than downloading it every time it appears.

Remember that linked graphics are the result of placing an <img> element within a set of <a> tags. For example:

```
<a href="index.html"><img src="home.gif" alt="home page
image" width="50" height="20" /></a>
```

Refer to Chapter 8 for more information on working with images.

### Using the alt Attribute

As you read earlier, you should provide alternate text-based links in addition to graphical links. You can do so by including an alt attribute in the <img> tag of the HTML code for the graphic. Repeating navigation options ensures that you meet the needs of a wide range of users. Some sites choose not to offer a text-based alternative, and this makes it difficult for users who cannot view graphics in their browsers. Figure 9-14 shows the navigation bar from the Barnes and Noble Web site with images turned on and off. Notice that even though images do not appear, the user

can still navigate the site because the <img> elements contain meaningful alt attribute values.

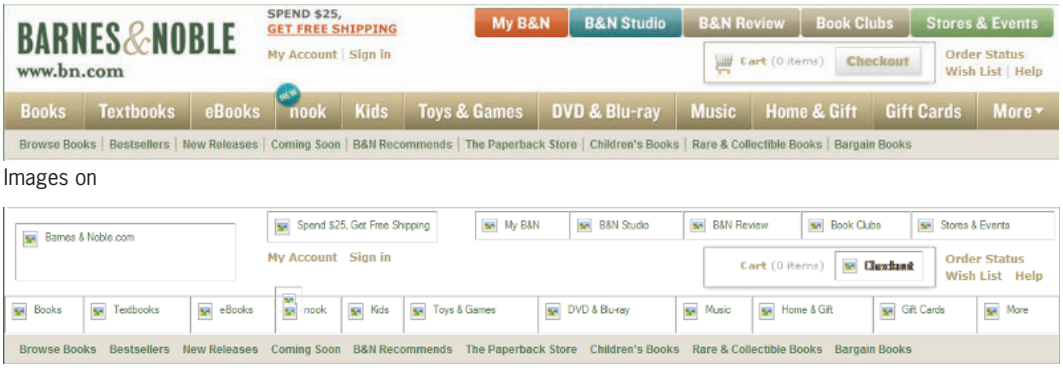


**Figure 9-14** Barnes and Noble Web site navigation bar with images on and off

The user finds navigation cues by reading the alt text and pointing to the image areas to find the clickable spots. Accessibility devices can use the alt attribute to provide navigation information as spoken content or in other media. The inclusion of alt attributes is of prime importance to the accessibility of your Web site.

Remember that the alt attribute is different from the title attribute. Alt is designed to provide the alternate text as shown above in Figure 9-14. The title attribute text is displayed in a ToolTip or ScreenTip (a pop-up window that appears when the user points to an object). You can read more about the title attribute in Chapter 8.

## Using Icons for Navigation

Figure 9-15 shows the navigation icons from the MapQuest Web site (*www.mapquest.com*). The text labeling on the icons points out one of the main problems with icons—not everyone agrees on their meaning. Especially with a worldwide audience, you never can be sure exactly how your audience will interpret your iconic graphics. This is why so many Web sites choose text-based links, and many that use icons, such as the MapQuest example in Figure 9-15, choose to include descriptive text with the icons.


**Figure 9-15** Using icons for navigation

No matter what types of graphics you choose as icons, make sure that your users understand their meaning. Test your navigation graphics on users in your target audience, and ask them to interpret the icons and directional graphics you want to use. The most obvious type of graphics to avoid are symbols that are culturally specific, especially hand gestures (such as thumbs up), which

may be misinterpreted in other cultures. Other graphics, such as directional arrows, or accepted representational graphics, are more likely to be interpreted correctly.

## Using Lists for Navigation

Refer to Chapter 5 for more information on list elements.

The HTML list elements are the preferred element for containing navigation links. Lists provide an easy way to create navigation that can be styled with CSS, are accessibility compatible, and work well even if a browser does not support CSS. If for some reason the CSS style rules are not applied, the navigation links will still be shown as a list.

The HTML list elements are commonly used to create bulleted and numbered lists. The following code shows conventional use of the <ul> and <li> elements to create a bulleted list. This list has an id *navlist* that will be used for applying style rules.

```
<ul id="navlist">
   <li><a href="index.html">Home</a></li>
   <li><a href="history.html">History</a></li>
   <li><a href="how.html">How it Works</a></li>
   <li><a href="clubs.html">Balloon Clubs</a></li>
   <li><a href="festivals.html">Festivals</a></li>
   <li><a href="rides.html">Where to Ride</a></li>
   <li><a href="faq.html">FAQ</a></li>
</ul>
```

This code results in the bulleted list shown in Figure 9-16.


**Figure 9-16** Basic unordered list

## Removing Default Padding and Margin

The bulleted list in Figure 9-16 has built in spacing that indents the list from the left side of the browser window. Depending on the browser, this built-in spacing is either applied using padding or margin. In most instances you will need to remove this default spacing before creating navigation lists. Set both the margin and padding properties to zero for the <ul> element, as shown in the following code, which selects a <ul> element with an id *navlist*.

```
ul#navlist {
   padding: 0;
   margin: 0;
}
```

## Removing Default Bullets

HTML lists come with built-in bullets, which are useful when you are creating standard item lists. When you are building lists for navigation, you can remove the default bullets with the list-style-type property. You can add this to the same style rule you use to remove the default padding and margin, as shown below:

```
ul#navlist {
   padding-left: 0;
   margin-left: 0;
   list-style-type: none;
}
```

The result of removing both the default bullets and indenting results in the list shown in Figure 9-17. This list is ready to be turned into either a horizontal or vertical navigation bar, as you will see in the next sections.


**Figure 9-17** List element with default indent and bullets removed

## Building Horizontal Navigation Bars

In a standard list element, the list items are block-level elements. A line break separates each item onto its own new line. To create a horizontal navigation bar using a list, you need to set each list item's display setting to *inline*, allowing the list to be displayed without line breaks. You can do this with the display property. The following style rule uses an id selector and descendant selection to select the <li> elements within a <ul> element with an id *navlist*. The style rule sets the display property to *inline*.

Refer to Chapter 6 for more information on the display property and Chapter 4 for more information on CSS selectors.

```
ul#navlist li{
   display: inline;
}
```

The horizontal navigation bar in Figure 9-18 is styled with three styles rules: one that styles the <ul> container, one that sets the <li> elements to inline, and a third that styles the <a> elements that create the hypertext links.

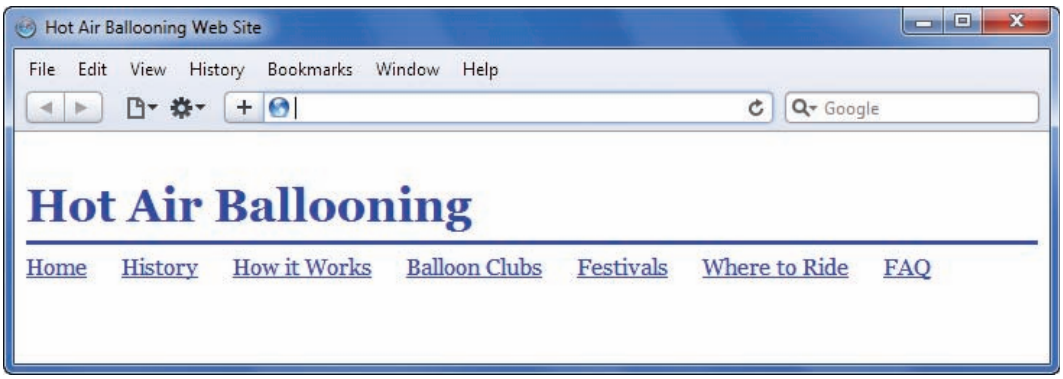


**Figure 9-18** Unordered list styled as horizontal navigation

The first style rule removes the default spacing and bullets from the unordered list:

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

The second style rule sets the <li> elements to inline display:

```
ul#navlist li{
   display: inline;
}
```

The third style rule styles the <a> elements that contain the link text. In this example, a right margin is added to each link to provide some white space between each link in the horizontal list.

```
ul#navlist a{
   margin-right: 20px;
   }
```

## Customizing the Horizontal Navigation Bar

Once you have created the basic list, you can customize it using different CSS style properties. You can remove underlining, add borders and background colors or images, and set space between buttons, all with a few style rules. For example, you can build on the basic horizontal navigation bar in Figure 9-18, which was created with these rules:

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

```
ul#navlist li{
   display: inline;
}

ul#navlist a{
   margin-right: 20px;
}
```

Add a border and background color to the <a> element that contains the link text, and set the right margin from 20px to 5px, bringing the link elements closer together.

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

ul#navlist li{
   display: inline;
}

ul#navlist a{
   margin-right: 5px;
   border: solid 1px
   background-color: #ccccff;
}
```

Add 5 pixels of top and bottom padding and 10 pixels of left and right padding. Set the text decoration to *none* to remove the underlining from the links.

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

ul#navlist li{
   display: inline;
}

ul#navlist a{
   margin-right: 5px;
   border: solid 1px red;
   background-color: #ccccff;
   padding: 5px 10px 5px 10px;
   text-decoration: none;}
```

Finally, add a top margin to the <ul> element to offset the entire navigation bar from the heading above.

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

ul#navlist li{
   display: inline;
}

ul#navlist a{
   margin-right: 5px;
   border: solid 1px;
   background-color: #ccccff;
   padding: 5px 10px 5px 10px;
   text-decoration: none;}
```

This code results in the enhanced navigation bar shown in Figure 9-19.

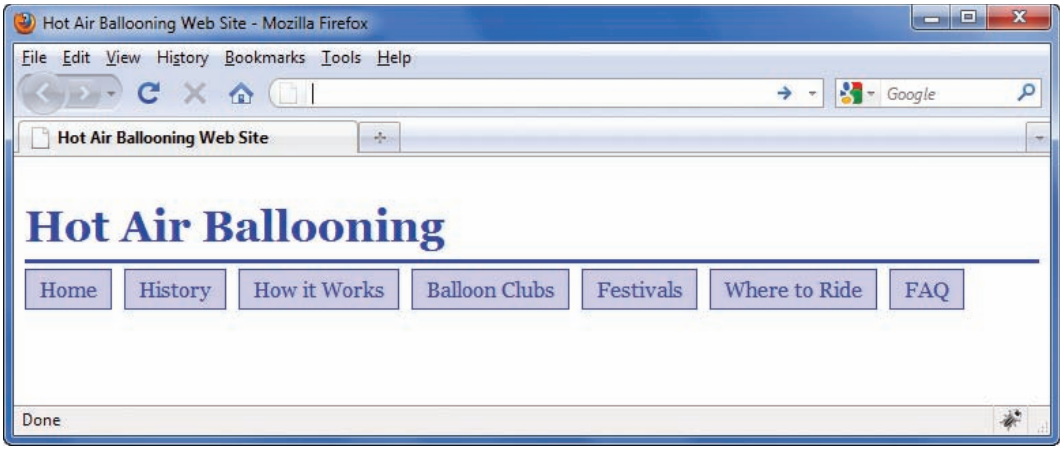


**Figure 9-19** Enhanced horizontal navigation bar

## Controlling Navigation Bar Width

Your horizontal navigation bar will wrap if a user makes their browser window small enough, as shown in Figure 9-20. To keep this from happening, add a width property to the <ul> element style rule, as shown in the following code:

```
ul#navlist {
   padding: 0;
   margin: 10px 0px 0px 0px;
   list-style-type: none;
   width: 700px;
}
```


**Figure 9-20** Fix this undesirable wrapping with the width property

## Controlling Navigation Button Width

In the previous example, the horizontal navigation bar buttons are based on the size of the text they contain. You may want to have all navigation buttons be the same width. To create same-size buttons, you will have to change the display type of the <a> elements to *block* to set a consistent width, and then float the boxes so they will align next to each other. The result is the navigation bar shown in Figure 9-21.

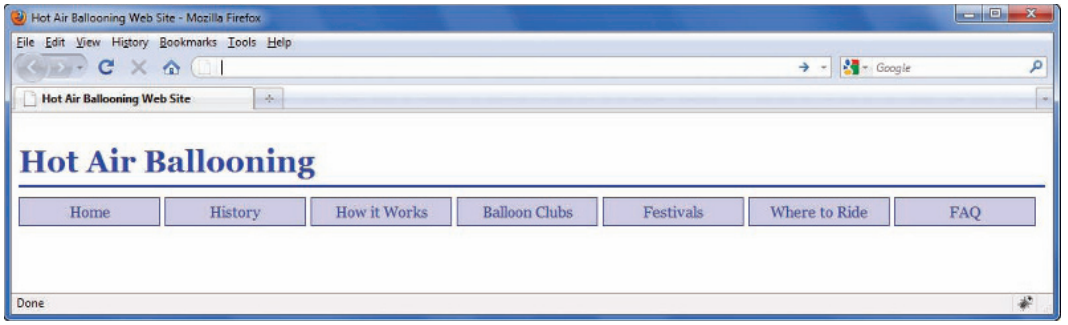


**Figure 9-21** Horizontal navigation bar with consistent button width

To accomplish this, add a float property to the <li> elements within the list:

```
ul#navlist {
   padding: 0;
```

```
   margin: 10px 0px 0px 0px;
   list-style-type: none;
}

ul#navlist li{
   display: inline;
   float: left;
}
```

Set properties for the <a> elements that contain the hypertext links. Set the display property to *block* so you can set a width for the buttons. Choose a width that will contain the longest piece of text in your link buttons, in this case 7em. Align the text to center with the text-align property.

```
ul#navlist a{
   margin-right: 5px;
   border: solid 1px red;
   background-color: #ccccff;
   padding: 5px 10px 5px 10px;
   text-decoration: none;
   display: block;
   width: 7em;
   text-align: center;
```

Ems are a good choice for navigation button width because they adapt to the user's font size. If the user chooses a larger or smaller default font for their browser, the navigation buttons resize accordingly.

## Building Vertical Navigation Bars

When you are building a vertical navigation bar, you can use a standard list structure without changing the display type as you did for a horizontal navigation bar. The common style of vertical navigation bars usually includes buttons that are the clickable links to different areas of a Web site. These can be styled in an endless variety of ways.

As you saw in the horizontal navigation bar, the <a> elements are contained within <li> elements to create the clickable hypertext links. Because <a> elements are inline by default, you will need to set each <a> element's display setting to *block*. This will let you create clickable buttons of any width you choose.

This example uses the same HTML list elements as you saw in the horizontal example. What is interesting about this example is that you can produce two very different results, in this case both a horizontal and a vertical navigation bar with the same HTML code, just by varying the CSS style rules.

```
<ul id="navlist">
   <li><a href="index.html">Home</a></li>
   <li><a href="history.html">History</a></li>
   <li><a href="how.html">How it Works</a></li>
   <li><a href="clubs.html">Balloon Clubs</a></li>
```

```
   <li><a href="festivals.html">Festivals</a></li>
   <li><a href="rides.html">Where to Ride</a></li>
   <li><a href="faq.html">FAQ</a></li>
</ul>
```

To create a vertical navigation bar, start by setting the margin and padding to zero and removing the default bullet:

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

Now style the <a> elements that reside within the <li> elements. Set the display to *block* for the links. Remove the underlining from the link text with the text-decoration property. Set a width of 140 pixels, and add a 2-pixel-wide solid blue (#0033cc) border.

```
ul#navlist li a {
   text-decoration: none;
   display: block;
   width: 140px;
   border: 2px solid #0033cc;
}
```

This style rule results in the incomplete navigation bar in Figure 9-22. You can see the basic structure, but it needs some more style rules to look better.

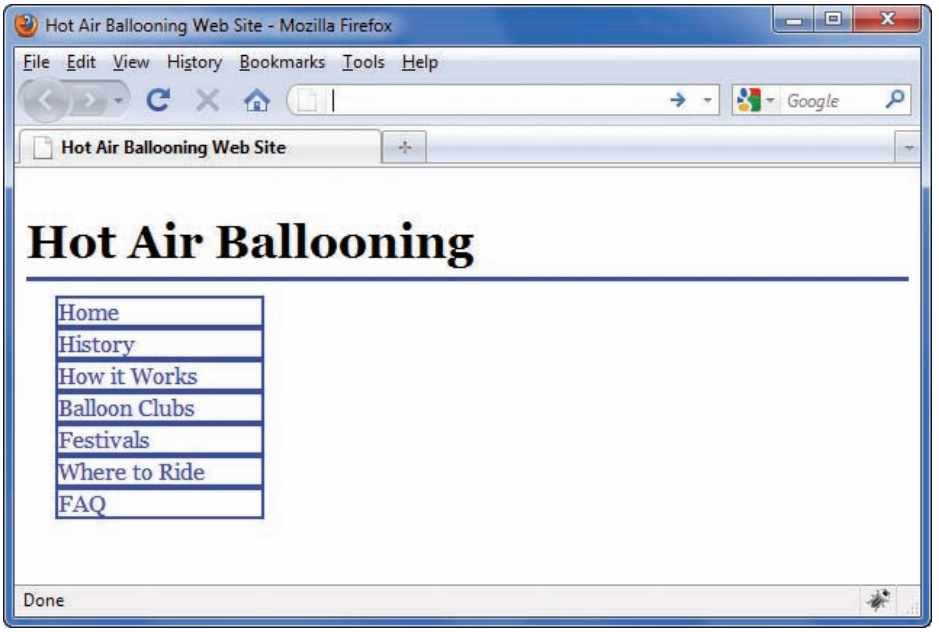


**Figure 9-22** Building the vertical navigation bar

To finish the navigation bar, add 5 pixels of padding to offset the link text from the borders of the button. Set the background color to a light blue (#ccccff), and change the text color to black (#000). Finally, add 5 pixels of top margin to separate the buttons vertically.

```
ul#navlist li a {
   text-decoration: none;
   display: block;
   width: 140px;
   border: 2px solid #0033cc;
   padding: 5px;
   background-color: #ccccff;
   color: #000;
   margin-top: 5px;
}
```

Adding this code results in the finished navigation bar shown in Figure 9-23.

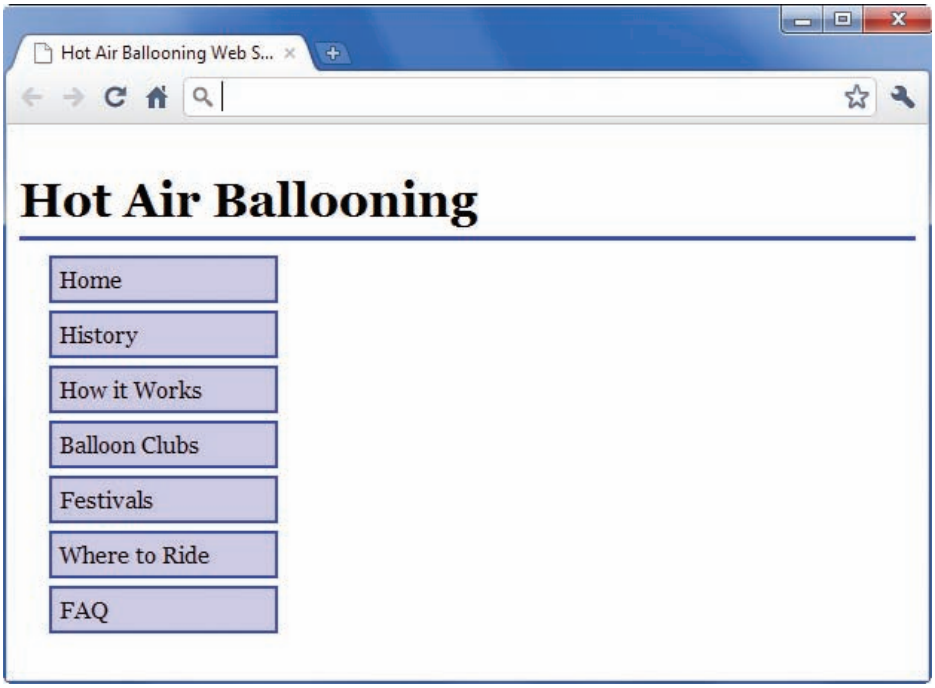


**Figure 9-23** Unordered list styled as vertical navigation

# Using Background Color and Graphics to Enhance Navigation

Refer to Chapter 8 for more information on using background graphics.

You can use background colors and graphics in a variety of ways to enhance your navigation. You can indicate location with a graphic or by changing a background color. You can create an interactive hover that changes a color or background when the user points to a navigation link.

## Indicating History

You can use the link pseudo-classes (described in Chapter 4) along with CSS background images to show a user where they have been on your Web site. Figure 9-24 shows a simple table of contents where the user's visited links are indicated with a red check mark. This keeps track of which pages the user has visited in the site.


**Figure 9-24** Indicating history with a background graphic

To create this effect, you must leave enough room in the background area of the element to display the image. In this example, the <a> element content needs to be moved in from the left margin by 30 pixels, which is accomplished by the following style rule. Note that this style rule affects only <a> elements with a *class="chapter"* attribute, so that other <a> elements on the page are not affected.

```
a.chapter {padding-left: 30px;}
```

The result of this rule is that 30 pixels of white space are added to the left of the text, leaving room for the check mark to show, as illustrated in Figure 9-25.

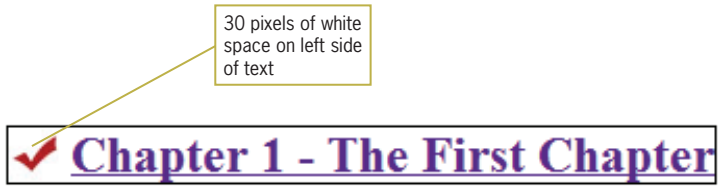


**Figure 9-25** Using padding-left to make room for the check mark

The selector in the following style rule applies the background graphic only to <a> elements that have a state of *visited* and a *class="chapter"* attribute.

```
a:visited.chapter { }
```

You can then specify the background graphic location, repetition, and positioning. The background image property points to the location of the file. The background-repeat property specifies that the image should only appear once. Finally, the background-position property states that the image is aligned horizontally to the left and vertically to the center height of the element.

```
a:visited.chapter {
   background-image: url(redcheck.jpg);
   background-repeat: no-repeat;
   background-position: left center;
}
```

The result of these styles is that the user will see a check mark next to any visited pages.

## Indicating Location

Location can be indicated by a change in text weight, text color, background color, or with an indicating graphic, which is commonly found on the Web. Figure 9-26 shows a horizontal navigation bar from the AIGA Web site with a background color indicating the current page.


**Figure 9-26** Navigation bar indicates current location

Indicating location can be as simple as using bold text instead of a link in a navigation bar on a page, as shown in Figure 9-27.

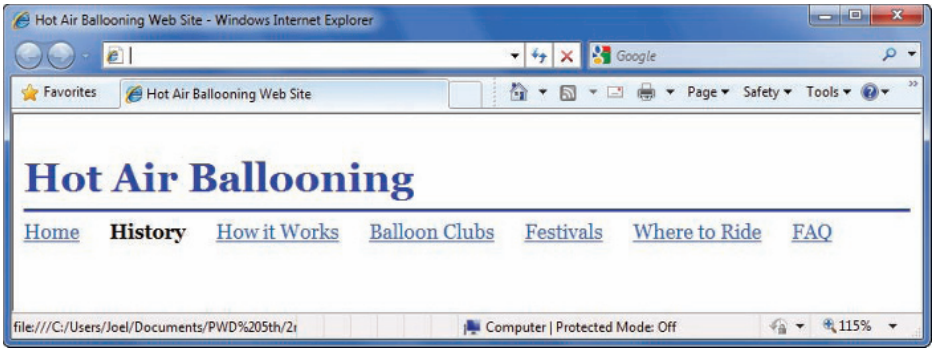


**Figure 9-27** Indicating location with bold text

This is easy to accomplish with a style named *current*, for example. This style will be applied to the link text for the current page. The style rule looks like the following:

```
#current {
   font-weight: bold;
}
```

The *current* style rule can then be applied with a  element, identifying the current page in the code for the navigation bar as shown:

```
<ul id="navlist">
   <li><a href="home.html">Home</a></li>
   <li><span id="current">History</span></li>
   <li><a href="how.html">How it Works</a></li>
   <li><a href="clubs.html">Balloon Clubs</a></li>
   <li><a href="festivals.html">Festivals</a></li>
   <li><a href="rides.html">Where to Ride</a></li>
   <li><a href="faq.html">FAQ</a></li>

</ul>
```

This same type of style rule can be used to change any number of characteristics for the current page text. For example, to change the background color for the current page text, use the background-color property as shown:

```
#current {
   background-color: #f90000;
}
```

# Creating Hover Rollovers

See Chapter 4 for more information on using the :hover pseudo-class.



You can use the CSS :hover pseudo-class with a variety of effects to add interactivity when users scroll over a list of navigation links or buttons. You can change text colors, background colors, and background images based on user actions.

## Changing Text Color and Background Color on Hover

Figure 9-28 shows an interactive hover that changes the text color and background color when the user points the mouse pointer at any of the links in the navigation bar.

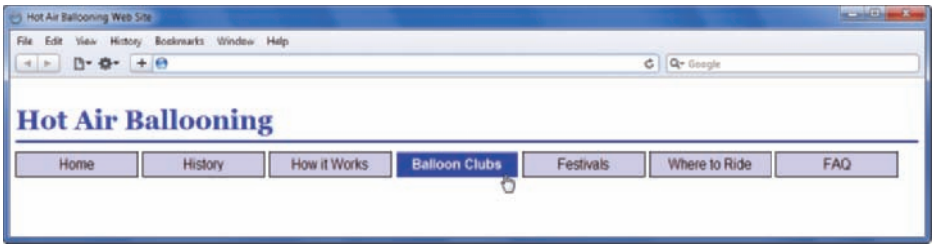


**Figure 9-28** Hovering the mouse changes link text and background color

Here are the style rules for the navigation bar. This is the same code you saw before in the horizontal navigation bar in Figure 9-21.

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

ul#navlist li{
   display: inline;
   float: left;
}

ul#navlist a{
   font-family: arial, sans-serif;
   font-weight: bold;
   margin-right: 5px;
   border: solid 1px;
   padding: 5px 10px 5px 10px;
   text-decoration: none;
   color: #000;
   background-color: #ccccff;
   width: 125px;
   text-align: center;
   display: block;
}
```

The hover effect is created with the following style rule. Note that this selects only <a> elements within the <ul> element that have an *id="navlist"* element; otherwise, it would affect all <a> elements on the page. When the user hovers the mouse over the link, the text color changes to white (#fff), and the background color changes to bright blue (#0033cc). The font-weight property makes the text bold.

```
ul#navlist a:hover {
   color: #fff;
   background-color: #0033cc;
   font-weight: bold;
}
```

## Changing Background Images on Hover

You can change background images as easily as changing background colors. For example, Figure 9-29 shows a navigation bar composed of three-dimensional button graphics. When the user hovers the pointer over a button, the button changes color from light blue to dark blue. The buttons are CSS background images that are displayed behind the link text and switch from the light blue button to the dark blue button based on the user action.

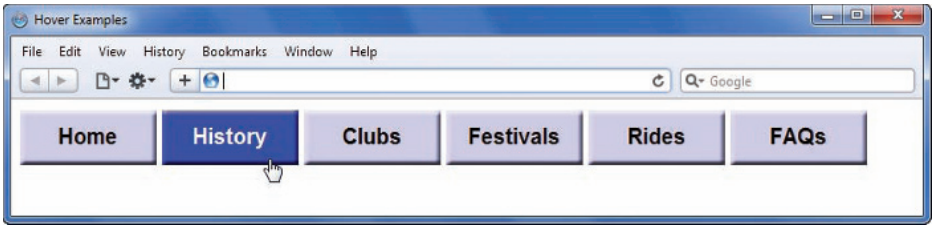


**Figure 9-29** Navigation bar with 3-D buttons

Figure 9-30 shows the two buttons used in this navigation bar.

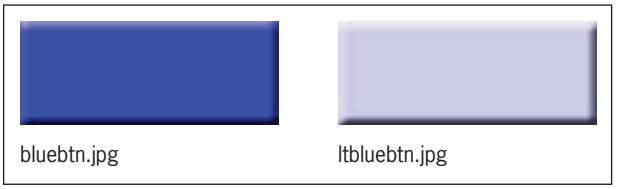


**Figure 9-30** Navigation bar background button images

Here are the style rules that make up the entire navigation bar. Like the example earlier, the <a> element style rule includes the

font characteristics for the link text, along with some padding and margin properties. The width and height of the <a> element matches the width and height of the button area, making the entire button clickable. Note the background image specifies ltbluebtn.jpg.

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

ul#navlist li{
   display: inline;
   float: left;
}

a.navbutton{
   font-family: arial;
   font-size: 1.25em;
   font-weight: bold;
   padding-top: 12px;
   margin-right: 5px;
   text-decoration: none;
   width: 125px;
   height: 50px;
   text-align: center;
   display: block;
   color: #000;
   background-image: url(ltbluebtn.jpg);
   background-repeat: no-repeat;

}
```

The hover effect is created with the following style rule. Note that this selects only <a> elements with a class="navbutton" attribute; otherwise, it would affect all <a> elements on the page. When the user hovers the pointer over the link, the background image changes from ltbluebtn.jpg to bluebtn.jpg.

```
a.navbutton:hover {
   background-image: url(bluebtn.jpg)
   color: #fff;
}
```

## Underlining on Hover

Many Web sites disable the default underlining of hypertext links. The CSS text-decoration property sets the value to *none*, which turns off hypertext linking as shown in the following code:

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

You can also use background colors to highlight links using the background-color property, as described in the Chapter 4 section titled "Using the :hover Pseudo-Class."

You can use the hover pseudo-class to turn the underlining on when the user points to the link. The style rule looks like this:

```
a:hover {text-decoration: underline;}
```

You can also create other linking effects by substituting a bottom border instead of the standard text underlining. In this style rule, the border-bottom property is used to display a dashed blue line when the user points at the link, as shown in Figure 9-31.

```
a:hover {border-bottom: dashed blue;}
```

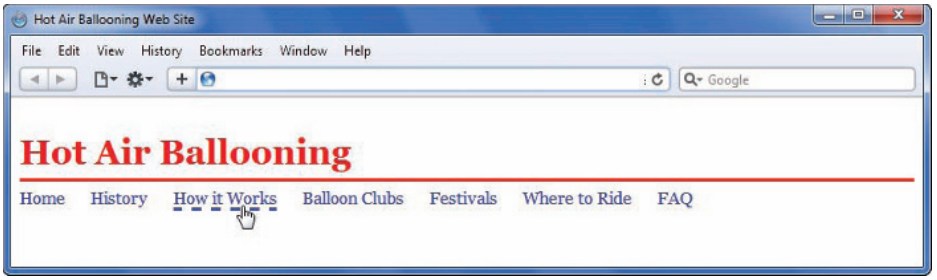


**Figure 9-31** Underlining a link with the :hover pseudo-class

## Chapter Summary

Usable navigation is the result of working with the power of hypertext and designing for your users' needs. Keep the following points in mind:

- Work from the user's point of view. Think about where users want to go within your Web site, and make it easy for them to get there.
- Add plenty of links to make all areas of your Web site quickly accessible to your users. Link to fragments as well as whole pages. Make it easy to get back to your navigation options.
- In addition to providing links, make sure you provide plenty of location cues to let users know where they are.
- Use text-based navigation bars to link users to other pages in your site. Use other text-based links to help users move through a long page of information or through a table of contents.
- Consider text as an alternative to graphical links. Every graphic adds to download time. When using graphics and icons as

navigational links, make sure users can interpret these links correctly by including text as part of the images. Also, be sure to use navigation icons consistently throughout your Web site to provide predictable cues for users and to minimize download time.

- Include alt values to your <img> tags to provide alternate navigation options for users.
- Use CSS to build attractive horizontal and vertical navigation bars using simple list elements.
- You can use background colors, text colors, and graphics to enhance navigation by indicating history or location.
- You can use the hover pseudo-class to add interactivity to navigation.

## Key Terms

**breadcrumb path**—A series of links, usually at the top of a Web page, that shows the user's location within the site hierarchy. Users can click any of the links in the path to move through the content structure.

**contextual link**—A link that allows users to jump to related ideas or cross-references by clicking the word or item that interests them. You can embed contextual links directly in your content by choosing the key terms and concepts you anticipate your users will want to follow.

**fragment identifier**—The use of the <a> element and name attribute to name a segment of an HTML file. You then can reference the fragment name in a hypertext link.

## Review Questions

1. List three advantages of linking by using text instead of graphics.

2. What four navigation questions should the user be able to answer?

3. List three types of navigation cues.

4. List three ways to control information overload when designing a Web site.

5. Explain why you would include both graphic and text-based links on a Web page.

6. List two navigation cues you can add to a text-based navigation bar.

7. Why is it best to make <a> the innermost element to a piece of text?

8. What <a> tag attribute is associated with fragment identifiers?

9. List two ways to break up lengthy HTML pages.

10. What character entity is useful as an invisible link destination?

11. What attribute do you use to make an <a> tag both a source and destination anchor?

12. How do you link to a fragment in an external file?

13. Page turners work best in what type of structure?

14. What are the benefits of contextual linking?

15. List two reasons for standardizing graphics.

16. What are the benefits of using navigation graphics?

17. What are the drawbacks of using navigation icons?

18. What are the benefits of using the alt attribute?

## Hands-On Projects

1. This book's Online Companion Web site contains all the HTML files for the sample Web site illustrated in Figure 9-2. Use these sample HTML files to build an alternate navigation scheme. Refer to the information structure illustrations in Chapter 3 (Figure 3-8 through

Figure 3-16) for examples of different navigation models. Choose a structure and code examples of usable navigation for the model.

2. In this project, you build a horizontal navigation bar. The code you add to the file in the following steps appears in blue.

   a. Copy the **horizontal nav bar.html** file from the Chapter 09 folder provided with your Data Files to the Chapter09 folder in your work folder. (Create the Chapter 09 folder, if necessary.)

   b. Start your text editor, and open the file **horizontal nav bar.html**. The HTML code is an unordered list with an id of *navlist.*

```
<ul id="navlist">
<li><a href=" ">Welcome</a></li>
<li><a href=" ">Services</a></li>
<li><a href=" ">Portfolio</a></li>
<li><a href=" ">About Us</a></li>
<li><a href=" ">Contact Us</a></li>
<li><a href=" ">FAQ</a></li>
</ul>
```

   c. Open the **horizontal nav bar.html** file in a browser. The list should look like Figure 9-32.

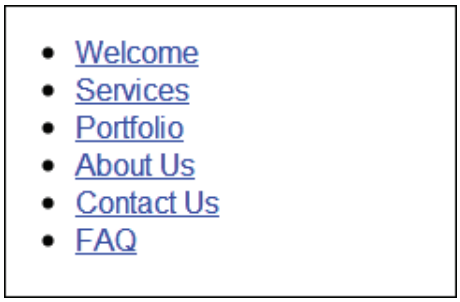


**Figure 9-32** Unordered list in the beginning horizontal nav bar file

   d. In the style section, create a selector for the navigation list:

```
ul#navlist { }
```

e. Add style properties to remove the default spacing and bullets from the unordered list:

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

f. Write another rule that selects the <li> elements and sets them to inline display:

```
ul#navlist li{
   display: inline;
}
```

g. Write a third style that selects the <a> elements containing the link text. Add a right margin to provide some white space between each link in the horizontal list:

```
ul#navlist a{
   margin-right: 20px;
   }
```

h. Save your file and view it in your browser. The finished navigation bar looks like Figure 9-33.

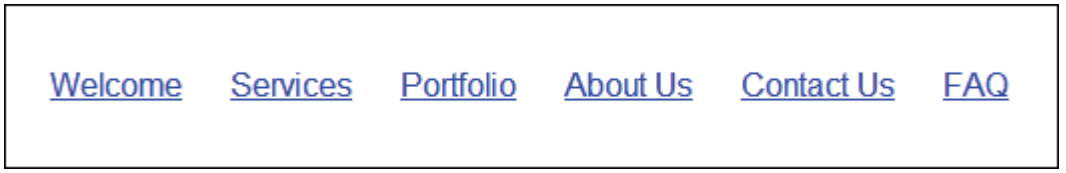


**Figure 9-33** Completed horizontal nav bar

3. In this project, you build a vertical navigation bar. The code you add to the file in the following steps appears in blue.

a. Copy the **vertical nav bar.html** file from the Chapter 09 folder provided with your Data Files to the Chapter09 folder in your work folder. (Create the Chapter09 folder, if necessary.)

b. Start your text editor, and open the file **vertical nav bar.html**. The HTML code is an unordered list with an id of *navlist.*

```
<ul id="navlist">
<li><a href=" ">Welcome</a></li>
<li><a href=" ">Services</a></li>
```

```
<li><a href=" ">Portfolio</a></li>
<li><a href=" ">About Us</a></li>
<li><a href=" ">Contact Us</a></li>
<li><a href=" ">FAQ</a></li>
</ul>
```

c. Open the **vertical nav bar.html** file in a browser. The list should look like Figure 9-34.

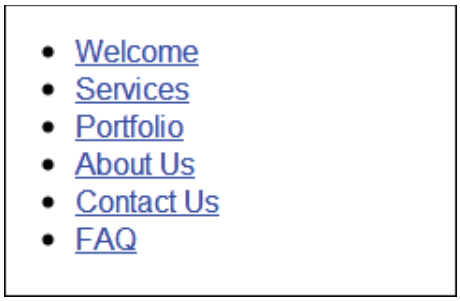


**Figure 9-34** Unordered list in the beginning vertical nav bar file

d. In the style section, create a selector for the navigation list:

```
ul#navlist { }
```

e. Add style properties to remove the default spacing and bullets from the unordered list:

```
ul#navlist {
   padding: 0;
   margin-left: 30px
   list-style-type: none;
}
```

f. Create a second rule that selects the <a> elements residing within the <li> elements:

```
ul#navlist li a { }
```

g. Add style rules that set the display to *block* for the links. Remove the underlining from the link text with the text-decoration property. Set a width of 140 pixels, and add a 2-pixel solid border.

```
ul#navlist li a {
   text-decoration: none;
   display: block;
   width: 140px;
   border: 2px solid;
}
```

h. Finish the navigation bar by adding 5 pixels of padding to offset the link text from the borders of the button. Also add 5 pixels of top margin to separate the buttons vertically.

```
ul#navlist li a {
   text-decoration: none;
   display: block;
   width: 140px;
   border: 2px solid;
   padding: 5px;
   margin-top: 5px;
}
```

i. Save your file and view it in your browser. The finished navigation bar looks like Figure 9-35.


**Figure 9-35** Completed vertical nav bar

4. Browse the Web and find a Web site that has a successful navigation design. Write a short summary of why the navigation is effective and how it fits the user's needs. Consider the following criteria:

   a. Are the linking text and images meaningful?

   b. Is it easy to access the site's features?

   c. Can you easily search for content?

   d. Is there a site map or other site-orienting feature?

5. Find an online shopping Web site.
   a. Examine the navigation options, and indicate whether you think the navigation adds to or detracts from the online shopping experience.
   b. Describe how to change the navigation to increase its effectiveness.

6. Find an online information resource likely to be used for research. Examine the navigation options, and describe how the navigation helps or hinders the user's information-searching process. Consider the following:
   a. How cluttered is the user interface? Does it deter finding information?
   b. Is navigation prominent or secondary to the page design? Does the user always know his or her location in the site? Is the linking text concise and easy to understand? Is the link destination easy to determine from the linking text?
   c. How deep is the structure of the site? How many clicks does it take to get to the desired information?

7. Browse the Web to find examples of Web sites that need better navigation options. Using examples from the Web site, describe how you would improve the navigation choices.

8. Browse the Web to find a Web site that uses more than one navigation method, and describe whether this benefits the Web site and why.

9. Find a site that illustrates a navigation method different from the ones described in this chapter. Describe the navigation method and state whether this benefits the Web site and why.

## Individual Case Project

Examine the flowchart you created for your Web site. Consider the requirements of both internal and external navigation. Create a revised flowchart that shows the variety of navigation options you are planning for the Web site.

Using your HTML editor, mark up examples of navigation bars for your content. Make sure your filenames are intact before you start coding. Save the various navigation bars as separate HTML files for later inclusion in your Web pages.

Plan the types of navigation graphics you want to create. Sketch page banners, navigation buttons, and related graphics. Find sources for navigation graphics. For example, you can use public domain (noncopyrighted) clip art collections on the Web for basic navigation arrows and other graphics.

## Team Case Project

Work as a team to examine and discuss the flowchart you created for your Web site. Consider the requirements of both internal and external navigation. Collaborating as a team, refine your work to create a revised flowchart that shows the variety of navigation options you are planning for the Web site. Make sure you have standardized your filenames and all team members agree on the naming conventions.

Work individually to create text-based navigation bars for your content. Use your HTML editor to mark up examples of navigation bars for site navigation. Save the various navigation bars as separate HTML files. Have a team meeting where all members can present their navigation schemes. Choose the best scheme or combine features from different schemes to create the navigation for the group's Web site.

Work together to plan the types of navigation graphics you want to create. Assign team members to complete the remaining tasks, which include sketching page banners, navigation buttons, and related graphics, as well as finding sources for navigation graphics. For example, you can use public domain (noncopyrighted) clip art collections on the Web for basic navigation arrows and other graphics.