CHAPTER 5

Delving into Images and Links

Although text is the dominant form of content on most websites, there are plenty of other types of media in use, most notably graphics. In this chapter, you will learn about the different types of graphics that are used on the web. You will learn how to display graphics on your web pages and to provide alternative text descriptions of graphics to support browsers that have problems working with graphics. You will also learn how to use graphics when setting up links and to use links as navigation controls that connect your pages together and to other web pages. In addition to all this you will learn how to work with audio and video and to display other types of media content.

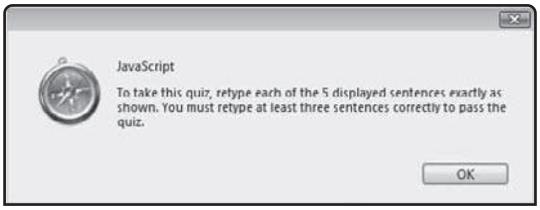
Specifically, you will learn how to:

- Use links to manage site navigation
- Use links to set up document downloads
- Use graphics when constructing links
- Embed and control the playback of audio and video
- Display PDF files within web pages

Project Preview: The (X)HTML Typing Quiz

In this chapter's web project, you will learn how to create an online quiz that evaluates the user's typing skills. Specifically, you will develop an online application that presents the users with five sentences and challenges them to retype those sentences exactly as they are shown. Any mistake in spelling, punctuation, or case will result in an error. To pass the quiz, users must retype at least three sentences correctly.

When loaded, the application begins by displaying the popup dialog windows shown in Figure 5.1.



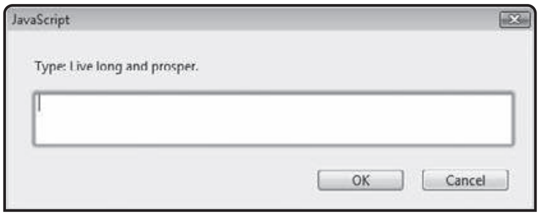
JavaScript

To take this quiz, retype each of the 5 displayed sentences exactly as shown. You must retype at least three sentences correctly to pass the quiz.

Figure 5.1

An example of the application's opening popup Window as seen using Apple Safari.

The user is presented with a brief explanation of the rules for taking the typing quiz and must click on OK to continue. In response, the first of five challenge sentences is displayed, as demonstrated in Figure 5.2.



JavaScript

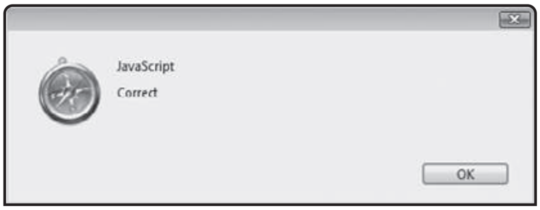
Type: Live long and prosper.

Figure 5.2

Sentences must be retyped in the popup dialog window's text field.

To submit an answer, the user must retype the sentence exactly as it is displayed in the window's text field and then click on the OK button. Mistyping the sentence or clicking on the window's Cancel button will result in an incorrect answer.

As demonstrated in Figure 5.3, the web application provides the user with immediate feedback after each answer is submitted.



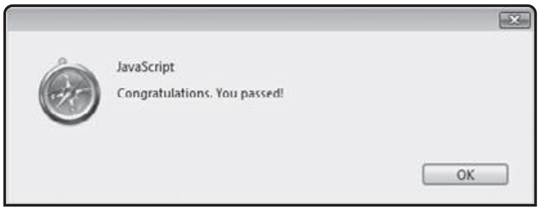
JavaScript

Correct

FIGURE 5.3

The user can keep track of progress by keeping count of the number of correctly retyped sentences.

At the end of the quiz, the application totals up the user's score and displays a message indicating whether the user passed or failed the quiz, as demonstrated in Figure 5.4.



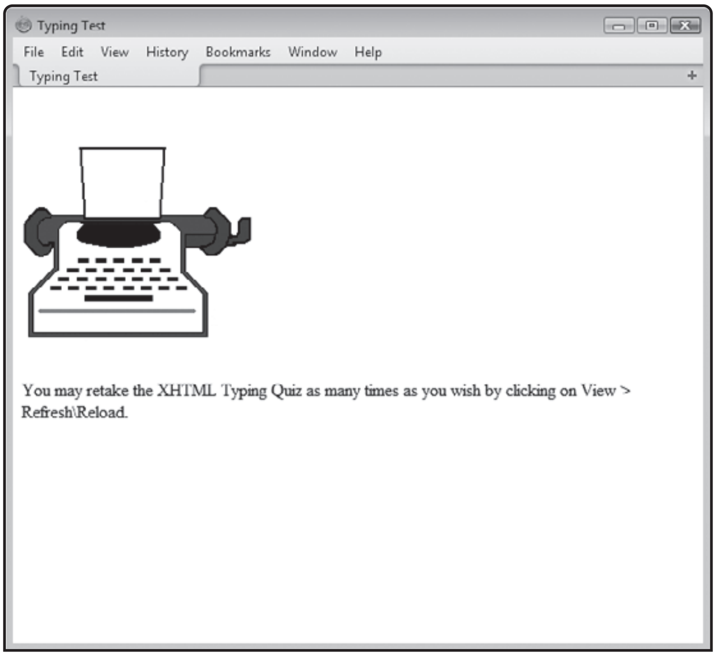
JavaScript

Congratulations. You passed!

FIGURE 5.4

The user has passed the typing quiz.

As soon as the user acknowledges her results by clicking on the OK button, the screen shown in Figure 5.5 is displayed, instructing the user how to restart the test.



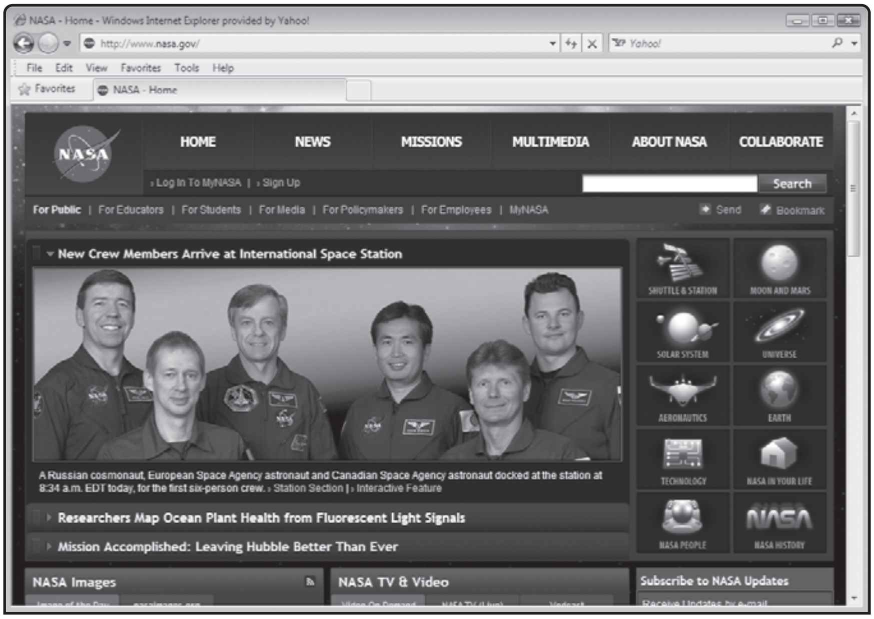
You may retake the XHTML Typing Quiz as many times as you wish by clicking on View > Refresh\Reload.

FIGURE 5.5

The user is permitted to retake the test as many times as is desired.

Let's Get Graphical

Up to this point in the book, focus has been almost exclusively on the use of text. However, sometimes ideas, messages, and concepts are better articulated through graphics like photographs, charts, graphs, drawings, maps, and so on. Many web developers use graphics to help establish a website's overall layout and presentation, using them to establish a consistent theme across their web pages. As demonstrated in Figure 5.6, many online websites carefully integrate their company's logo and branding into their website.



New Crew Members Arrive at International Space Station

A Russian cosmonaut, European Space Agency astronaut and Canadian Space Agency astronaut docked at the station at 8:34 a.m. EDT today, for the first six-person crew. › Station Section | › Interactive Feature

Researchers Map Ocean Plant Health from Fluorescent Light Signals

Mission Accomplished: Leaving Hubble Better Than Ever

Figure 5.6

An example of how graphics can be used to create a visually powerful and theme-oriented web page.

To be effective, graphics must be properly used; otherwise, they can backfire on you, resulting in web pages that look overloaded, convoluted, and unattractive. XHTML lets you work with graphics using either of two elements, the `img` or `object` element. Of these two options, the `img` element is by far the most commonly used. The `img` element is an inline element that can be used to work with all types of image files.

Image Types

There are many different types of graphics. Unfortunately, not all graphics are equally supported by all operating systems. As a result, it is best to stick with the universally supported graphics formats listed here:

- **GIF (Graphics Interchange Format).** Best used for images with less than 256 colors but that require great detail.
- **JPEG (Joint Photographic Experts Group).** Best used for photos and other graphics files that contain a larger range of colors.
- **PNG (Portable Network Graphics).** Best used for photos and other graphics files that contain a larger range of colors.

Any good graphics editor will support all three of these graphic formats and will allow you to convert files from one format to another. All three of these graphic formats are designed to compress the size of their files, making them well suited for the Internet where bandwidth is always a concern. JPEG files reduce the size of graphic files by reducing the number of pixels used to represent the image. The result is a loss of image quality. JPEGs do, however, retain a great deal of color information making them appropriate for storing photographs and other high-color images.

Unlike JPEG files, GIF files are limited to 256 colors but are capable of rendering great detail. Rather than sacrifice pixels, GIF files reduce the number of colors. Because they retain all pixels, GIF files are good for storing detailed line art, maps, graphs, charts, and other low-color graphics. GIF files provide two features that make them unique when compared to JPEGs. First, GIF files can have transparent backgrounds, meaning that if you overlay an image on top of another image, the bottom image will become a background for the GIF file. You can also create animated GIF files. These types of GIF files are actually made up of two or more frames that are rotated, giving the impression that the GIF file is in motion.

PNG files are a relatively new graphic file format. Like GIF files, PNG files retain all pixel data. PNG files support transparency. However, unlike GIF files, PNG files are not limited to 256 colors. In fact, PNG files can support 24-bit color, allowing them to rival JPEG files, though PNG files tend to be a little larger than their JPEG counterparts.

As you can see, all three of these graphic file types have their advantages and disadvantages and you will probably end up using different combinations of all three. The important thing is that you avoid using other graphic file types.

Storing Graphic Files Externally

Unlike text, images displayed on web pages are external content. They are not embedded in the web document. Instead, they reside in external files that are then referenced from within the web document. Web browsers render images in two steps. First, the web document's markup must be downloaded and then any external image files are downloaded.

Working with the img Element

The `img` element is an inline element, and as such must be embedded within a block-level element to be used. In order to generate well-formed pages, you must include the `src` and `alt` attributes. The `src` attribute is used to specify the URL location of the graphic file. The `alt` attribute provides an alternative text string that is displayed in the event the browser is unable to display the graphic file. The `alt` attribute allows you to specify up to 1,024 characters and should be used to provide descriptions that provide visitors with an effective understanding of the image and what it depicts. The following example demonstrates the use of the `img` element to display a graphic file named fishing.jpg, which is located in the same folder as the web document. An alternative description has also been provided.

```
<img src = "fishing.jpg" alt = "Picture of a little boy fishing" />
```

Specifying Graphic File Dimensions

Because web browsers download and render images in a two-step process, documents with many images may take a while to fully download and render, especially for users with slow Internet connections. As a result, users may begin reading page content while the images on the document are still being downloaded. The web browser will reserve space for each image file defined on a web document. However, since images vary in terms of width and height, there is no guarantee that the browser will allocate the correct amount of space required to display the image unless you provide it with information on the dimensions of each graphic. You can do this by adding the `width` and `height` attributes to the `img` element, as demonstrated in the following example.

```
<!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" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Missing Graphics Demo</title>
  </head>

  <body>
    <div>
      <img src = "fishing.jpg" width = "356" height = "267"
        alt = "Picture of a little boy fishing" />
      <img src = "teapot.jpg" width = "356" height = "267"
        alt = "Picture of a teapot" />
      <img src = "cats.jpg" width = "356" height = "267"
```

```
        alt = "Picture of two cats" />
    </div>
  </body>

</html>
```

As you can see, the `width` and `height` attributes have been included as part of all three of the web document's `image` elements. As a result, web browsers that support the display of graphics will reserve the proper amount of space for each graphic file. Figure 5.7 shows the web page that is created when this example is loaded and rendered using Apple Safari.

HINT

By default, all images appear in their actual size when rendered and displayed. If you specify reduced `width` and `height` attribute settings, it should look okay when rendered. However, to conserve bandwidth you should resize the graphics using your favorite graphic application to the size you want them to be.

If, on the other hand, you specify `width` and `height` attribute values that are larger than the graphic being download, the resulting image may appear choppy. Again, you will be better served editing your graphics and assigning the right width and height before you use them on your web pages. Check out Appendix B for information on several good graphic applications.

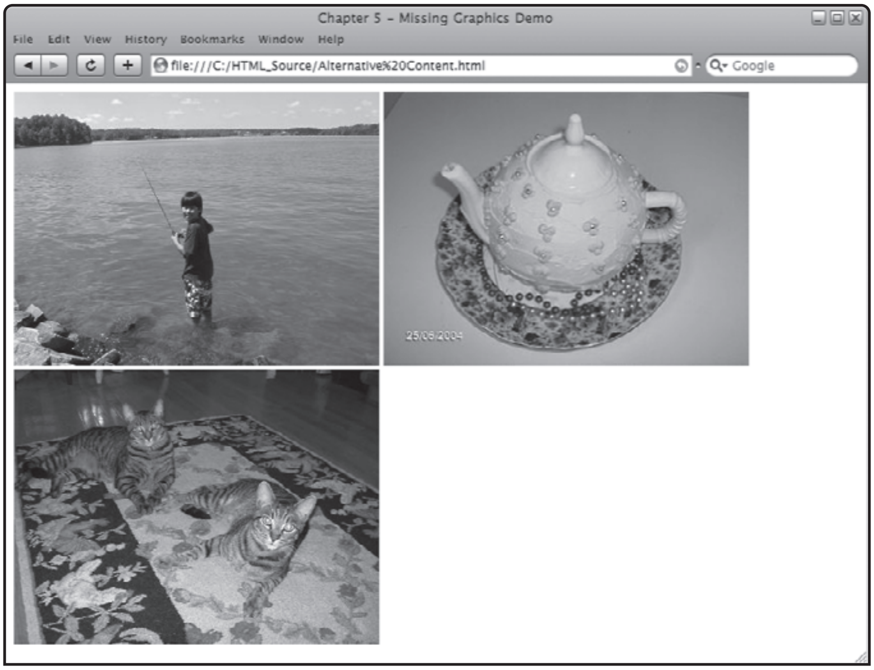

Chapter 5 – Missing Graphics Demo
File Edit View History Bookmarks Window Help
file:///C:/HTML_Source/Alternative%20Content.html
Google
25/06/2004

FIGURE 5.7

An example of how the web page looks when rendered by Apple Safari.

Figure 5.8 shows an example of what users will see in the event the web browser is unable to display the specific images.

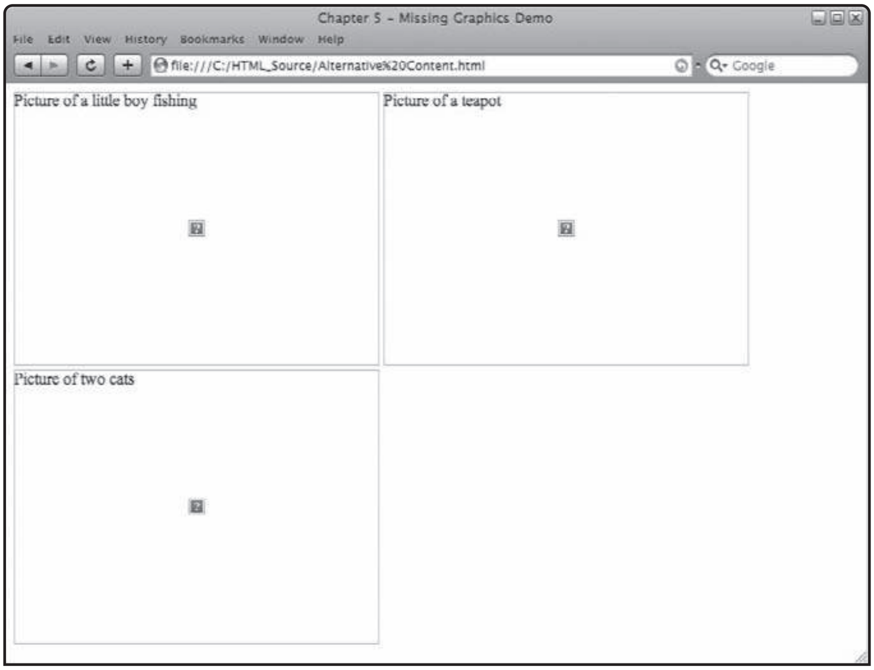

Chapter 5 – Missing Graphics Demo
File Edit View History Bookmarks Window Help
file:///C:/HTML_Source/Alternative%20Content.html
Google
Picture of a little boy fishing
Picture of a teapot
Picture of two cats

FIGURE 5.8 A demonstration of how alternative content is displayed in the event the graphics cannot be displayed.

There can be any number of reasons why a browser is unable to display graphic files. The browser may be a text-only browser. The graphic files may have been renamed, moved, or deleted. Regardless, as Figure 5.8 demonstrates, by including the required `alt` attribute you can still provide visitors with an idea of what they are missing.

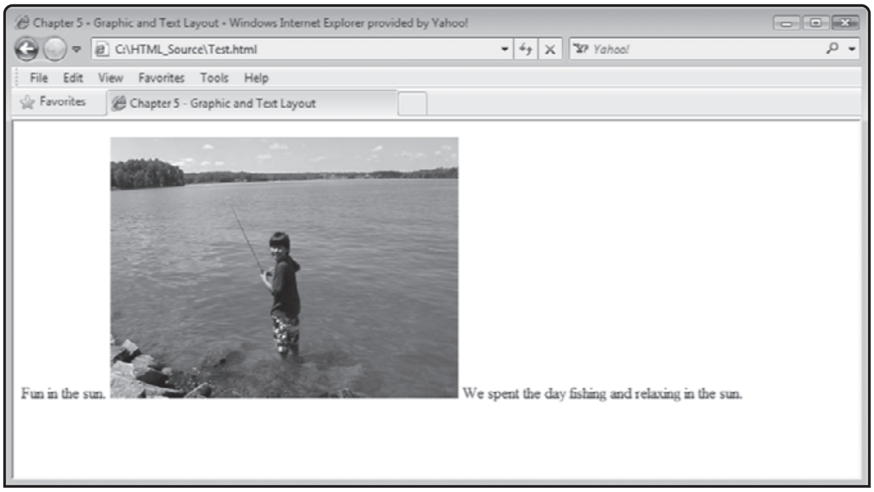
HINT To create well-formed web documents, you must include the `alt` attribute as part of every `img` element. However, since not all images are meant to serve as content, there may be images that don't convey any information at all and whose only purpose is to be aesthetically pleasing. For these types of graphics, you should assign an empty string (e.g., `alt = ""`) when specifying an `img` element `alt` attribute.

Controlling How Text and Graphics Are Displayed

By default, web browsers will display images alongside text. To demonstrate this, take a look at the following example.

```
<!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" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Graphic and Text Layout</title>
  </head>

  <body>
    <p>
      Fun in the sun.
      <img src = "fishing.jpg" width = "356" height = "267"
        alt = "Picture of a little boy fishing" />
      We spent the day fishing and relaxing in the sun.
    </p>
  </body>

</html>
```

Here, text and an `img` element have all been placed within a `p` element. Figure 5.9 shows how the web browser will render this missed content on the resulting web page.


Chapter 5 - Graphic and Text Layout - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
File Edit View Favorites Tools Help
Favorites
Chapter 5 - Graphic and Text Layout

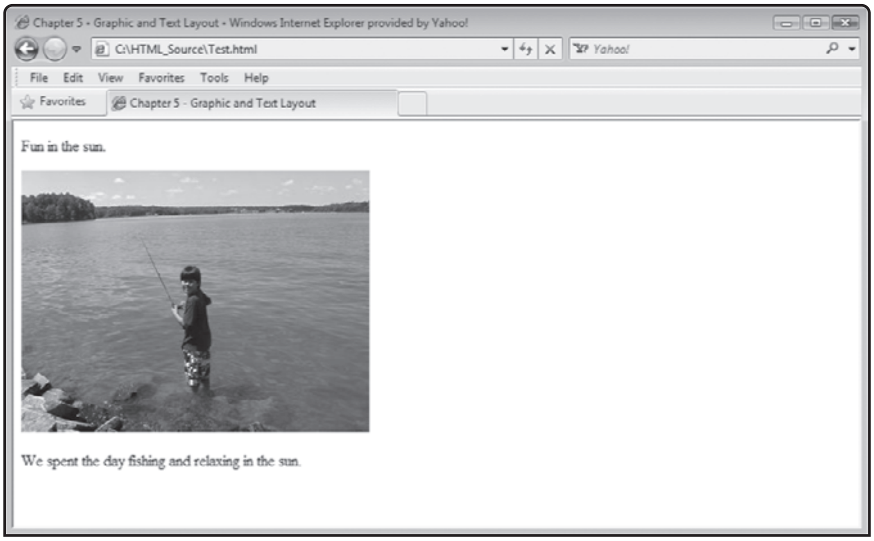
Fun in the sun. We spent the day fishing and relaxing in the sun.

FIGURE 5.9
Web browsers automatically align graphics and text.

If you want to separate the display of text and graphics, you can do so by enclosing them in their own respective block-level elements, as demonstrated in the following example.

```
<!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" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Graphic and Text Layout</title>
  </head>

  <body>
    <p>Fun in the sun.</p>
    <p><img src = "fishing.jpg" width = "356" height = "267"
      alt = "Picture of a little boy fishing" /></p>
    <p> We spent the day fishing and relaxing in the sun.</p>
  </body>

</html>
```

Figure 5.10 shows how the block-level format changes have affected the way the content of the resulting web page looks when it is rendered by the browser.


Chapter 5 - Graphic and Text Layout - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
File Edit View Favorites Tools Help
Favorites
Chapter 5 - Graphic and Text Layout

Fun in the sun.

We spent the day fishing and relaxing in the sun.

FIGURE 5.10

Using block-level elements to separate graphics and text.

Using CSS, you can exercise precise control over how text and graphics are displayed on your web pages. CSS is covered in Chapters 7 and 8.

Connecting Things Together with Links

Links, also known as hyperlinks, are used to connect things together on the Internet. They are in fact the glue that binds everything together to create the world wide web as we know it. Web browsers generally display links as underlined text. If you move the mouse pointer over a link it usually changes into a different shape, often a hand with a pointing finger. Click on the link and the web browser will navigate to the link's specified URL. Links can be set up for any of the following scenarios.

- To establish a link to another page.
- To establish a link to a location within the same page.
- To other types of documents in order to facilitate file downloads.
- To email addresses partially automating email generation.

Creating Links

Links are built using the anchor (`a`) element, which anchors a target URL to a location on a web page. The anchor element is an inline element, so to use it you must embed it within a block-level element, as demonstrated here:

```
<p><a href = "http://www.apple.com">Click Me</a></p>
```

Here, the anchor element's `href` attribute is used to specify the URL of the target file. The text (`Click Me`) embedded within the opening `<a>` tag and closing `</a>` tags is displayed in the browser. The link is activated when the user clicks on this text.

Setting Up Text Links

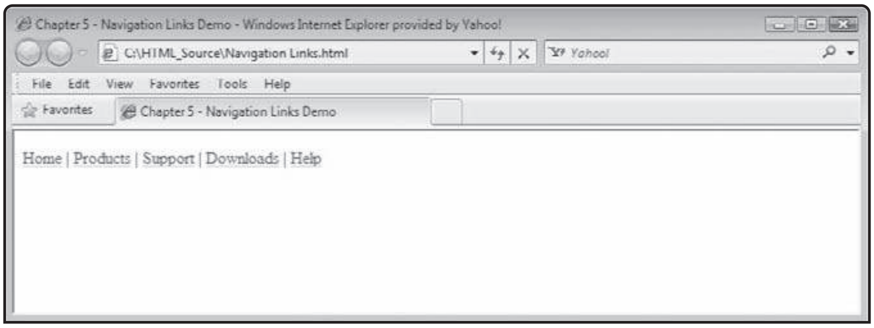
The most common use of links is to link to another external web document. To set up a link to an external document, you must know the document's full URL, as demonstrated in the following example:

```
<p><a href = "http://www.microsoft.com">Visit Microsoft</a></p>
```

Here, a link has been set up to the Microsoft website. When clicked, the web browser will leave the web page containing the link and load Microsoft's main web page. Web developers also use links as the basis for facilitating navigation between web documents on their own websites, as demonstrated in the following example.

```
<!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" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Navigation Links Demo</title>
  </head>

  <body>
    <p>
      <a href = "home.html">Home</a> |
      <a href = "products.html">Products</a> |
      <a href = "support.html">Support</a> |
      <a href = "downloads.html">Downloads</a> |
      <a href = "help.html">Help</a>
    </p>
  </body>

</html>
```

Here, five links have been defined that together form a text-based navigation bar that is often placed at the top and bottom of web pages to provide visitors with a means of navigating between web pages. Because these links are to local web pages that reside within the same folder as the current web page, relative URLs have been used. Figure 5.11 shows an example of the collection of links that are displayed when this web document is loaded into the browser.


Chapter 5 - Navigation Links Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Navigation Links.html
Yahoo!
File Edit View Favorites Tools Help
Favorites
Chapter 5 - Navigation Links Demo

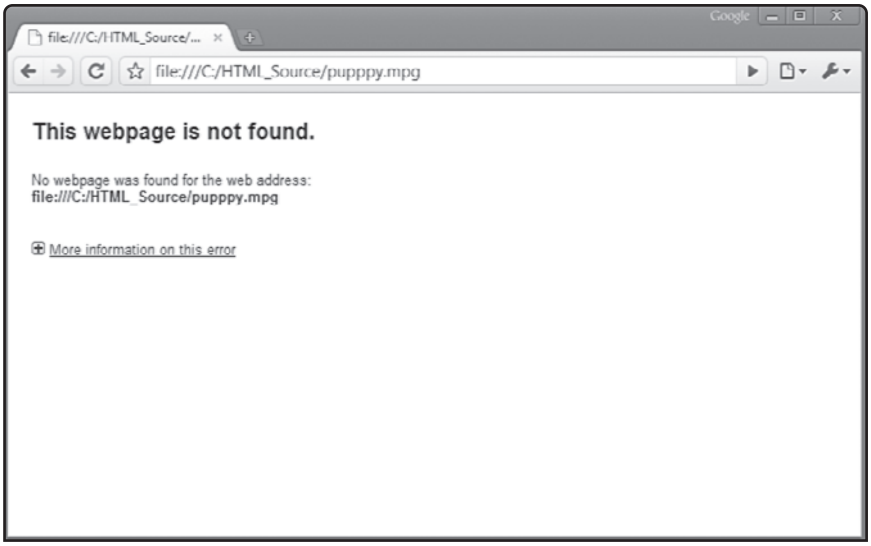
Home | Products | Support | Downloads | Help

FIGURE 5.11

Using links to create text-based navigation controls.

TRAP

One of the problems with links is that things on the Internet tend to change a lot. Any links that you set up to external pages can be broken at any time if their author deletes, renames, or moves them. Likewise, if you rearrange things within your own website, you run the risk of forgetting to update your site's internal links. Broken links result in errors like the one shown in Figure 5.12, which is the last thing you want the people that visit your website to see when they click on your links.



This webpage is not found.

No webpage was found for the web address:
file:///C:/HTML_Source/pupppy.mpg

More information on this error

FIGURE 5.12

An example of a broken link as viewed using Google Chrome.

Fortunately, there are a number of very good free and low-cost links checker web services and applications available that you can use to keep an eye on the status of all your website's links. Read Appendix B to learn more about them.

Setting Up Graphics Links

So far all of the links we've discussed have been text-based, but as the following example shows, you can substitute graphics in place of text. This allows you to create things like graphics menus and custom button controls.

```
<!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" lang="en" xml:lang="en">
```

```
  <head>

    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Downloading the Puppy Video Demo</title>
  </head>

  <body>

    <div>
      <a href = "help.html" target = "_blank"><img src = "help.gif"
        alt = "Click to access help" /></a>
    </div>

  </body>

</html>
```

Here, a graphic image (a black button labeled Help) is displayed as shown in Figure 5.13. When the user moves the mouse pointer over the button, the shape of the pointer changes. When clicked, a new web document is opened in its own browser window. Note that by default, links load target web pages into the current browser window. However, by adding the target attribute, which is discussed in the next section, to the link, you override this behavior by instructing the browser to open a new browser window.

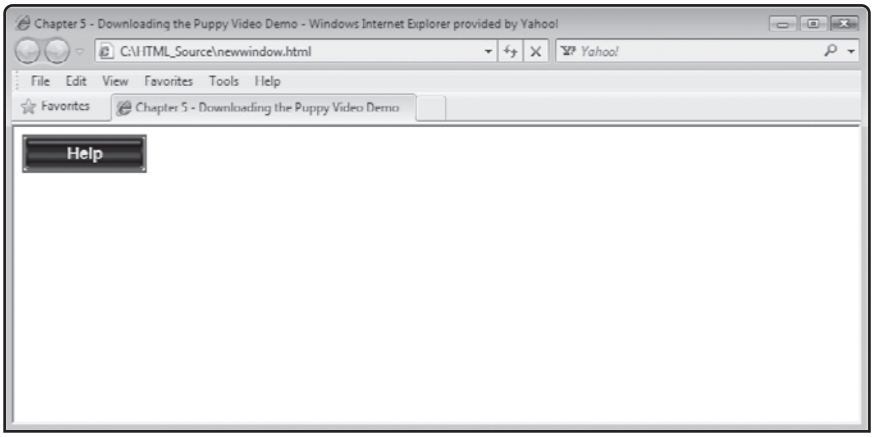

Chapter 5 - Downloading the Puppy Video Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\newwindow.html
Yahoo!
File Edit View Favorites Tools Help
Favorites
Chapter 5 - Downloading the Puppy Video Demo
Help

FIGURE 5.13

Configuring a graphic image to serve a link.

Don't Let Links Send Your Visitors Away

By default, a clicked link sends your visitors away from your website to the web page specified in the link. However, successful websites don't get that way by sending visitors away, but by keeping them around as much as possible. One way of providing your visitors with helpful links without sending them away is to create links that when clicked open a new window for the target page. This way, your visitors don't have to leave your website to explore one of your links.

The trick to creating links that open a new window is to include the anchor element's `target` attribute when defining a link and to assign a value of `_blank` to the attribute. An example of how to set up this type of link is shown here:

```
<!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" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Opening a Link in a New Window</title>
  </head>

  <body>
    <p>
      <p><a href = http://www.apple.com target = "_blank">Click Me</a></p>
    </p>
  </body>

</html>
```

Figure 5.14 shows the result that occurs when you run this example and click on the page's link.

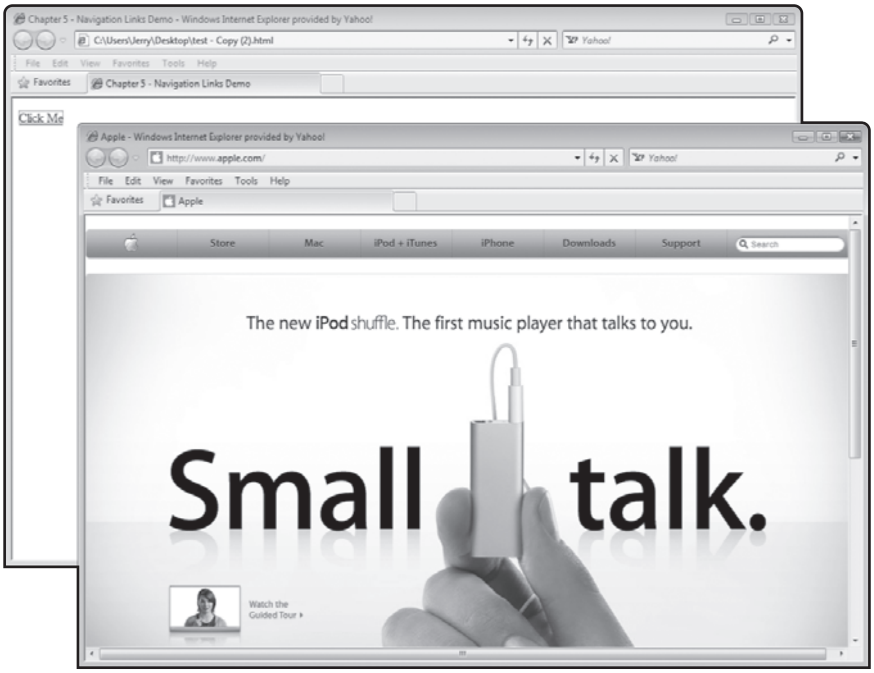

Chapter 5 - Navigation Links Demo - Windows Internet Explorer provided by Yahoo!
C:\Users\Jerry\Desktop\test - Copy (2).html
Click Me
Apple - Windows Internet Explorer provided by Yahoo!
http://www.apple.com/
Store
Mac
iPod + iTunes
iPhone
Downloads
Support
The new iPod shuffle. The first music player that talks to you.
Small talk.
Watch the Guided Tour

FIGURE 5.14

Keep your visitors by opening links in their own window.

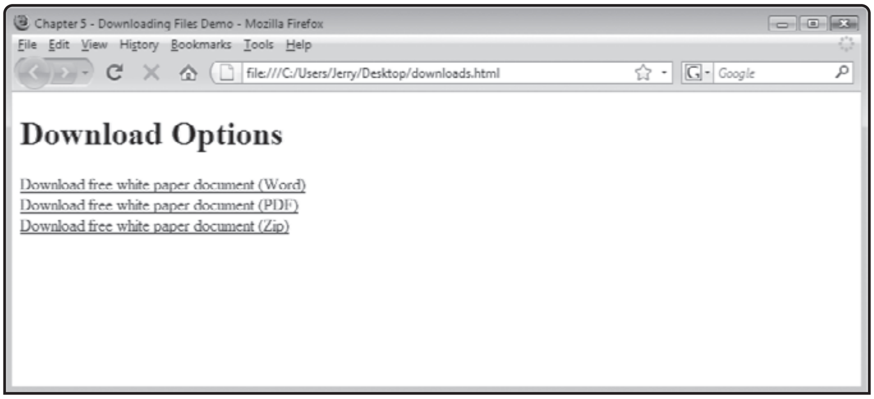
Using Links to Set Up Document Downloads

You can set up links to point to any type of file. This includes files like Microsoft Office documents, PDF files, and Zip archive documents. Since web browsers are not built to work with these types of files, a couple of things may happen. First, if the browser has a plug-in that can handle the document, which is typically the case for PDF files, it may open and display the file. Once displayed, visitors can view or save the PDF. For other types of files, the browser may display a window that allows your visitors to download and save the files.

The following example demonstrates how to set up a web page that displays links to three different files.

```
<!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" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Downloading Files Demo</title>
  </head>
  <body>
    <h1>Download Options</h1>
    <div>
      <a href = "WhitePaper.doc">
         Download free white paper document (Word)</a><br />
      <a href = "WhitePaper.pdf">
         Download free white paper document (PDF)</a><br />
      <a href = "WhitePaper.zip">
         Download free white paper document (Zip)</a>
    </div>
  </body>

</html>
```

Figure 5.15 shows how the resulting web page looks when loaded. Using this example as a model, you can create a "Downloads" page on your website where you might post all kinds of free files.


Chapter 5 - Downloading Files Demo - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Users/Jerry/Desktop/downloads.html
Google

Download Options

Download free white paper document (Word)
Download free white paper document (PDF)
Download free white paper document (Zip)

Figure 5.15
Making documents available for download.

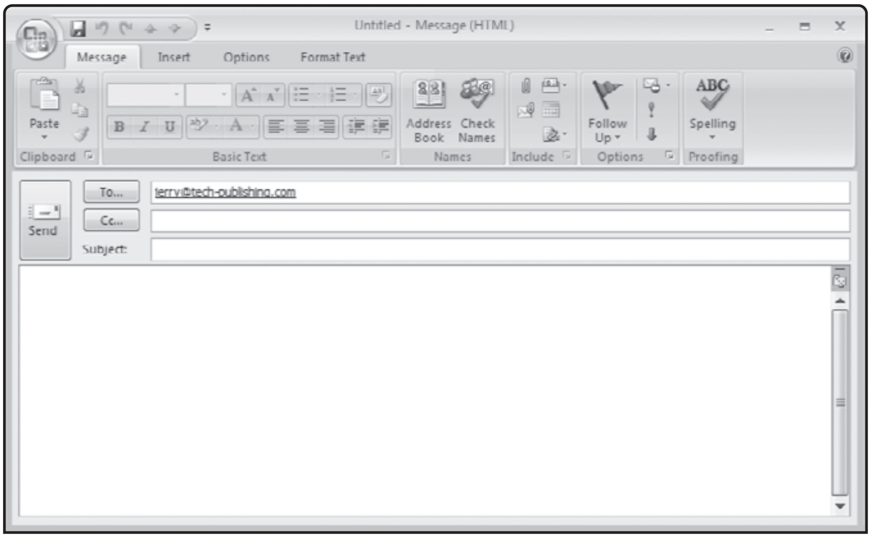
Using Links to Facilitate Emailing

Have you ever visited a website that displayed a link labeled Contact Us that when clicked automatically opened your email application and created a new email addressed to the website? Perhaps the title of the email was also included? If you want, you can use a link to replicate this same technique on your own web pages, as demonstrated here:

```
<!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" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Email Demo</title>
  </head>

  <body>
    <p>
      <a href="mailto:jerry@tech-publishing.com">Contact Us</a>
    </p>
  </body>

</html>
```

When clicked, this page's link will automatically start the user's default email application and copy the email address specified by the `href` attribute, less the required `mailto:` keyword into the application's To field. As a result, all the user has to do is supply a subject and message text and then click on Send to submit the email. Figure 5.16 shows an example.



To... terry@tech-publishing.com

FIGURE 5.16

Use a link to make it easy for your visitors to send you an email.

TRAP

Unfortunately, spammers have found that a good way of collecting email addresses is to collect them from web pages that use links to generate emails. So, unless you want to risk making it easier for spammers to get a hold of your email address, you might want to think hard before using this technique and instead use a form to collect visitor input. Forms are discussed in Chapter 6.

Other Forms of Content

As you have no doubt witnessed, websites are capable of displaying all kinds of content other than text and graphics. This includes content like audio and video. In order to work with this type of content you need to become familiar with the `object` element. In addition to audio and video, this element lets you include objects like java applets, ActiveX, Flash, and PDF in your web pages.

The key to working with video and audio in XHTML 1.0 Strict is to master use of the `object` element. The `object` element is an inline element. The `object` element supports a number of attributes that are used to define the media you are working with. These attributes include:

- **data.** Specifies the object's URL.
- **type.** Specifies the object's MIME type.
- **width.** Sets the object's width in pixels or as a percentage of its parent element.
- **height.** Sets the object's height in pixels or as a percentage of its parent element.

Integrating Video as Content

In order to use the `object` elements to embed and play audio and video files within your web documents, you must use the `object` element in conjunction with the `param` element. The `param` element must be nested within `object` elements in order to provide additional information needed to control interaction with the media, as demonstrated in the following example.

```
<!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" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Playing a MPG Video</title>
  </head>
  <body>
    <div>
      <object data = "puppy.mpg" type = "video/mpeg" width = "600"
        height = "480">
        <param name = "src" value = "puppy.mpg" />
        <param name = "autoplay" value = "true" />
      </object>
    </div>
  </body>

</html>
```

Here a video named puppy.mpg has been added to a web page using the `object` element. Two `param` elements were added that specify the name and location of the mpg file and instruct the browser to automatically start the playback of the video when the web page has finished loading. Figure 5.17 shows an example of the video as it is being played using the Internet Explorer browser.

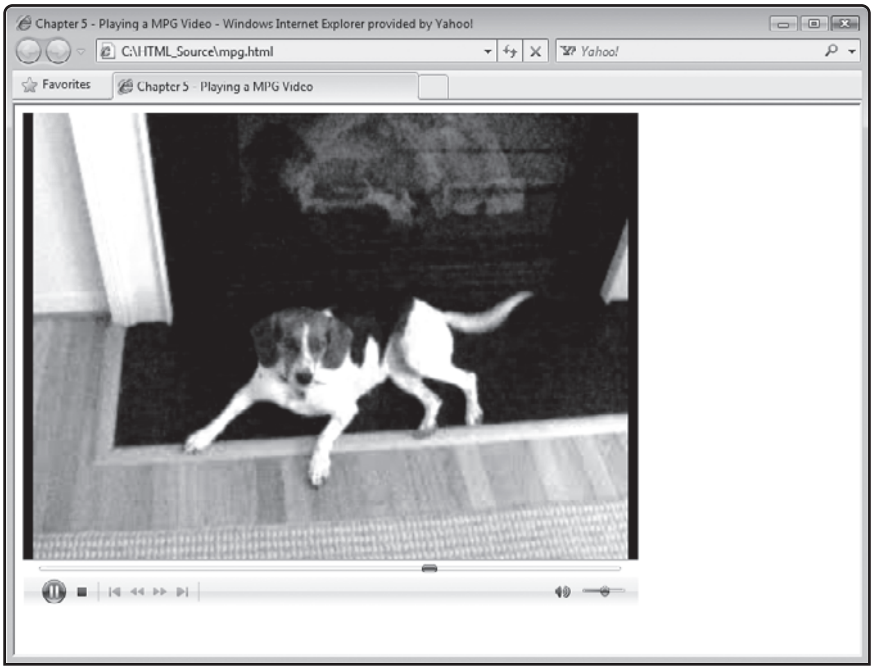

Chapter 5 - Playing a MPG Video - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\mpg.html
Yahoo!
Favorites
Chapter 5 - Playing a MPG Video

FIGURE 5.17

An example of how to play a video on your web page.

Adding Audio Playback to Your Web Pages

The playback of audio is not much different than the playback of video. The following example demonstrates how to display a control that manages audio playback.

```
<!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" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Playing an audio file</title>
  </head>

  <body>
    <div>
      <object type="audio/x-wav" data="test.wav" width="250" height="30">
        <param name="src" value="test.wav" />
        <param name="autoplay" value="true" />
        <param name="autoStart" value="1" />
      </object>
    </div>
  </body>
</html>
```

As you can see, a wave file named test.wav will be played. The `object` element's `type` attribute has been set to `audio/x-wav` and the `data` attribute has been set to `test.wav`. The dimensions of the playback control have also been specified. Next, three `param` elements are used to specify the location of the wave file and instruct the browser to initiate playback of the wave file.

Figure 5.18 shows how the resulting web page looks when this document is loaded using the opera web browser.

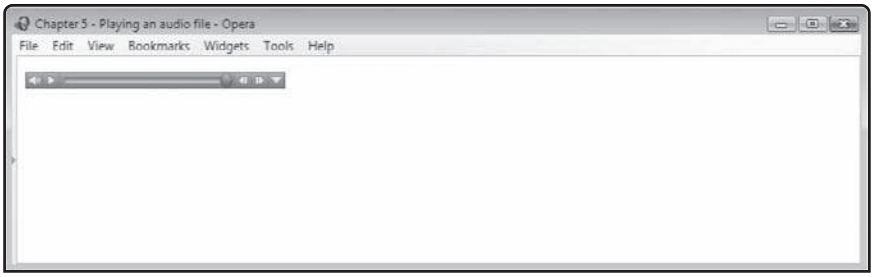

Chapter 5 - Playing an audio file - Opera
File Edit View Bookmarks Widgets Tools Help

FIGURE 5.18

An example of how to play an audio file on your web page.

If you prefer, you can remove the display of the playback control from the web page by setting the `object` element's `width` and `height` attributes to `0`, as demonstrated in the following example.

```
<!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" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Playing an audio file</title>
  </head>

  <body>
    <div>
      <object type="audio/x-wav" data="test.wav" width="0" height="0">
        <param name="src" value="test.wav" />
        <param name="autoplay" value="true" />
        <param name="autoStart" value="1" />
      </object>
    </div>
  </body>

</html>
```

Displaying PDF Documents

In addition to video and audio content, you can add other types of content to your web documents using the `object` element. For example, the following web document shows how to use the `object` element to embed the display of a PDF document as content within a web page.

```
<!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" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 5 - Displaying a PDF Document</title>
  </head>

  <body>
    <div>
      <object data="test.pdf" type="application/pdf" width="600"
        height="480">
      </object>
    </div>
  </body>

</html>
```

Figure 5.19 shows an example of the output that is produced when this example is rendered using Google's Chrome browser.

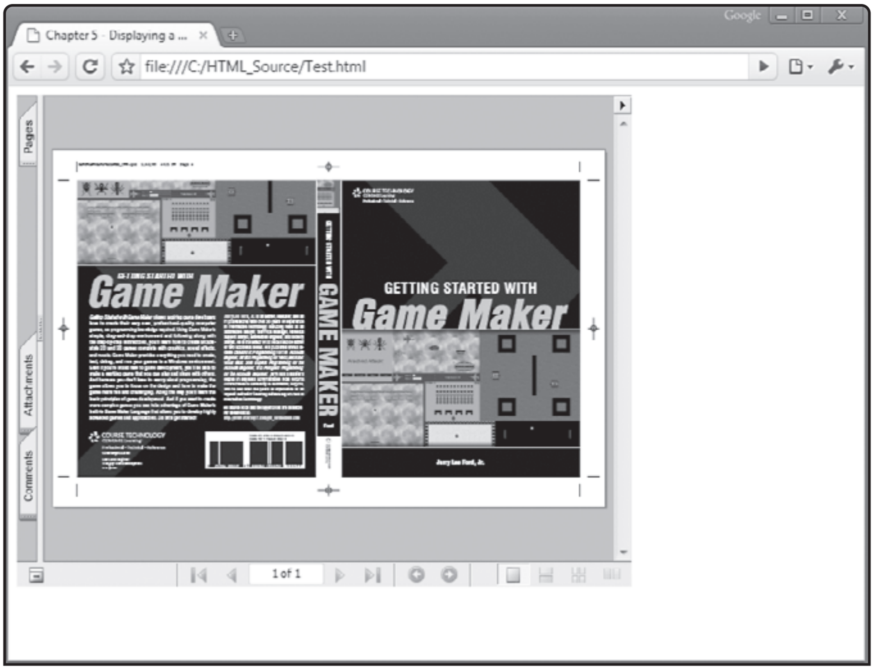


FIGURE 5.19

Displaying a DPF file as part of the content in a web page.

Back to the (X)HTML Typing Quiz

Alright, now it is time to return your attention to this chapter's project, the (X)HTML Typing Quiz. This web application presents the user with five sentences and challenges him to retype them exactly as shown. The user can take as much time as needed to retype each sentence but must retype it exactly as shown. Any change in spelling, case, or punctuation will result in an error. To pass the quiz the user must retype at least three sentences correctly.

Designing the Application

The development of this web application will be performed in four steps, as outlined here:

1. Create a new XHTML document.
2. Develop the document's markup.
3. Develop the document's JavaScript.
4. Load and test the XHTML page.

Step 1: Creating a New XHTML Document

This application consists of an XHTML document and a graphics file named typewriter.jpg. The first step in the creation of this project is the creation of the application's web document. Do so using your preferred code or text editor and then save and name the file typingquiz.html. Next, open your web browser and go to http://www.courseptr.com/downloads and download the source code for this project, including the project's graphic file. Place a copy of the typewriter.jpg file in the same folder as the project's XHTML file.

Step 2: Developing the Document's Markup

The second step in the development of the Typing Quiz application is to begin putting together the web document's markup. Begin by adding the following elements to the typingquiz.html file.

```
<!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" lang="en" xml:lang="en">

  <head>

  </head>

  <body>

  </body>

</html>
```

With the web document's core elements in place, let's ensure that the document is well formed by adding the following elements to its `head` section.

```
<meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
<title>Typing Quiz</title>
```

To wrap up your work on the document's markup, add the following statements to the `body` section.

```
<div><img src = "typewriter.gif" width = "222" height = "236"
  alt = "Picture of an old style manual typewriter" /></div>

<p>You may retake the (X)HTML Typing Quiz as many times as
    you wish by clicking on View > Refresh\Reload.</p>
```

As you can see, the `img` element is used to embed an image file named typewriter.jpg in the web document. The image file's `width` and `height` are set and an alternative text string is provided. Note that since the `img` element is an inline element, it must be enclosed within a block-level element, in this case the `div` element. Lastly, a paragraph has been added that provides instructions for taking the quiz.

Step 3: Creating the Document's Script

In order to manage the administration of the quiz and its interaction with the user, the web document includes a JavaScript. This script will manage the display of the web application's popup dialog windows and the analysis of each of the user's typed answers. To begin work on this script, add the following statements to the web document's `head` section.

```
<script type = "text/javascript">
<!-- Start hiding JavaScript statements

// End hiding JavaScript statements -->
</script>
```

The next step in the development of the script file is to declare its variables and to assign their initial values. To do this, add the following statements to the script file.

```
var noCorrect = 0;

var s1 = "Live long and prosper.";
var s2 = "In the end there can be only one.";
var s3 = "Indeed, perhaps today is a good day to die.";
var s4 = "If I had to choose, I would say that it is good to be ";
```

```
    s4 = s4 + "loved but better to be feared.";
var s5 = "Ask not what your country can do for you but what you ";
    s5 = s5 + "can do for your country.";

var intro = "To take this quiz, retype each of the 5 displayed " +
            "sentences exactly as shown. You must retype at " +
            "least three sentences correctly to pass the quiz."
```

The rest of the script's statements, shown next, should be added to the end of the script file.

```
alert(intro);

challenge = prompt("Type: " + s1, "");
if (challenge == s1) {
  alert("Correct");
  noCorrect = noCorrect + 1
} else {
  alert("Incorrect");
}

challenge = prompt("Type: " + s2, "");
if (challenge == s2) {
  alert("Correct");
  noCorrect = noCorrect + 1
} else {
  alert("Incorrect");
}

challenge = prompt("Type: " + s3, "");
if (challenge == s3) {
  alert("Correct");
  noCorrect = noCorrect + 1
} else {
  alert("Incorrect");
}

challenge = prompt("Type: " + s4, "");
if (challenge == s4) {
  alert("Correct");
  noCorrect = noCorrect + 1
} else {
  alert("Incorrect");
}

challenge = prompt("Type: " + s5, "");
if (challenge == s5) {
  alert("Correct");
  noCorrect = noCorrect + 1
} else {
  alert("Incorrect");
}

if (noCorrect > 2) {
  alert("Congratulations. You passed!");
} else {
  alert("Sorry but you failed.");
}
```

Without getting too much into the details of the script's programming logic, what these statements do is display a series of seven popup dialog windows that show the application's instructions, each of its five challenge sentences, and the user's final result.

HINT To learn more about JavaScript, make sure that you review Chapter 9.

The Finished HTML Document

Okay, assuming you have been following along carefully with all of the steps that have been outlined, your copy of the Typing Quiz should be ready for testing. To make sure that you have assembled everything correctly, take a look at the following example, which shows what the finished document should look like.

```
<!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" lang="en" xml:lang="en">

  <head>
```

```
<meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
<title>Typing Test</title>

<script type = "text/javascript">
<!-- Start hiding JavaScript statements

  var noCorrect = 0;

  var s1 = "Live long and prosper.";
  var s2 = "In the end there can be only one.";
  var s3 = "Indeed, perhaps today is a good day to die.";
  var s4 = "If I had to choose, I would say that it is good to be ";
      s4 = s4 + "loved but better to be feared.";
  var s5 = "Ask not what your country can do for you but what you ";
      s5 = s5 + "can do for your country.";

  var intro = "To take this quiz, retype each of the 5 displayed " +
              "sentences exactly as shown. You must retype at " +
              "least three sentences correctly to pass the quiz."

  alert(intro);

  challenge = prompt("Type: " + s1, "");
  if (challenge == s1) {
     alert("Correct");
     noCorrect = noCorrect + 1
  } else {
     alert("Incorrect");
  }

  challenge = prompt("Type: " + s2, "");
  if (challenge == s2) {
     alert("Correct");
     noCorrect = noCorrect + 1
  } else {
     alert("Incorrect");
  }
```

```
    challenge = prompt("Type: " + s3, "");
    if (challenge == s3) {
       alert("Correct");
       noCorrect = noCorrect + 1
    } else {
       alert("Incorrect");
    }

    challenge = prompt("Type: " + s4, "");
    if (challenge == s4) {
       alert("Correct");
       noCorrect = noCorrect + 1
    } else {
       alert("Incorrect");
    }

    challenge = prompt("Type: " + s5, "");
    if (challenge == s5) {
       alert("Correct");
       noCorrect = noCorrect + 1
    } else {
       alert("Incorrect");
    }

    if (noCorrect > 2) {
      alert("Congratulations. You passed!");
    } else {
      alert("Sorry but you failed.");
    }

  // End hiding JavaScript statements -->
  </script>

</head>

<body>

  <div><img src = "typewriter.gif" width = "222" height = "236"
```

```
      alt = "Picture of an old style manual typewriter" /></div>

    <p>You may retake the (X)HTML Typing Quiz as many times as
       you wish by clicking on View > Refresh\Reload.</p>


  </body>

</html>
```

Step 4: Loading and Testing the Typing Quiz

To test your new web application, start up you favorite web browser and use it to load the typingquiz.html file by executing the following steps.

1. Click on the browser's File menu and select the Open File command. This displays a standard file open dialog.
2. Navigate to the folder where you stored the web page and select it.
3. Click on the Open button.

If all goes as it should, continue testing your application using one or two other web browsers. You might want to validate that it is well formed by visiting thhp://validator.w3.org.

Summary

This chapter provided a good overview of how to integrate graphics into your web pages. You learned about JPEG, GIF, and PNG files and their advantages and disadvantages. You learned about the importance of providing alternative content for graphics, and you learned how to create graphic links, set up site navigation, and to set up document downloads. On top of all this, this chapter showed you how to work with the `object` element and how to use it to seamlessly integrate audio and video into your web documents. You even learned how to embed the display of PDF files as part of web page content. In addition, you learned how to create the Typing Quiz web application.

Challenges

1. As currently written, the Typing Quiz displays a somewhat bland looking graphic of a typewriter when the web page is rendered in the browser. Consider taking steps to spruce things up a bit by replacing this image with one that is a bit more interesting.
2. Consider adding a link to the bottom of the page that points back to your main website; perhaps mention that more interesting things await there.
3. If you feel up to the challenge, try to add to the text to make it more difficult.