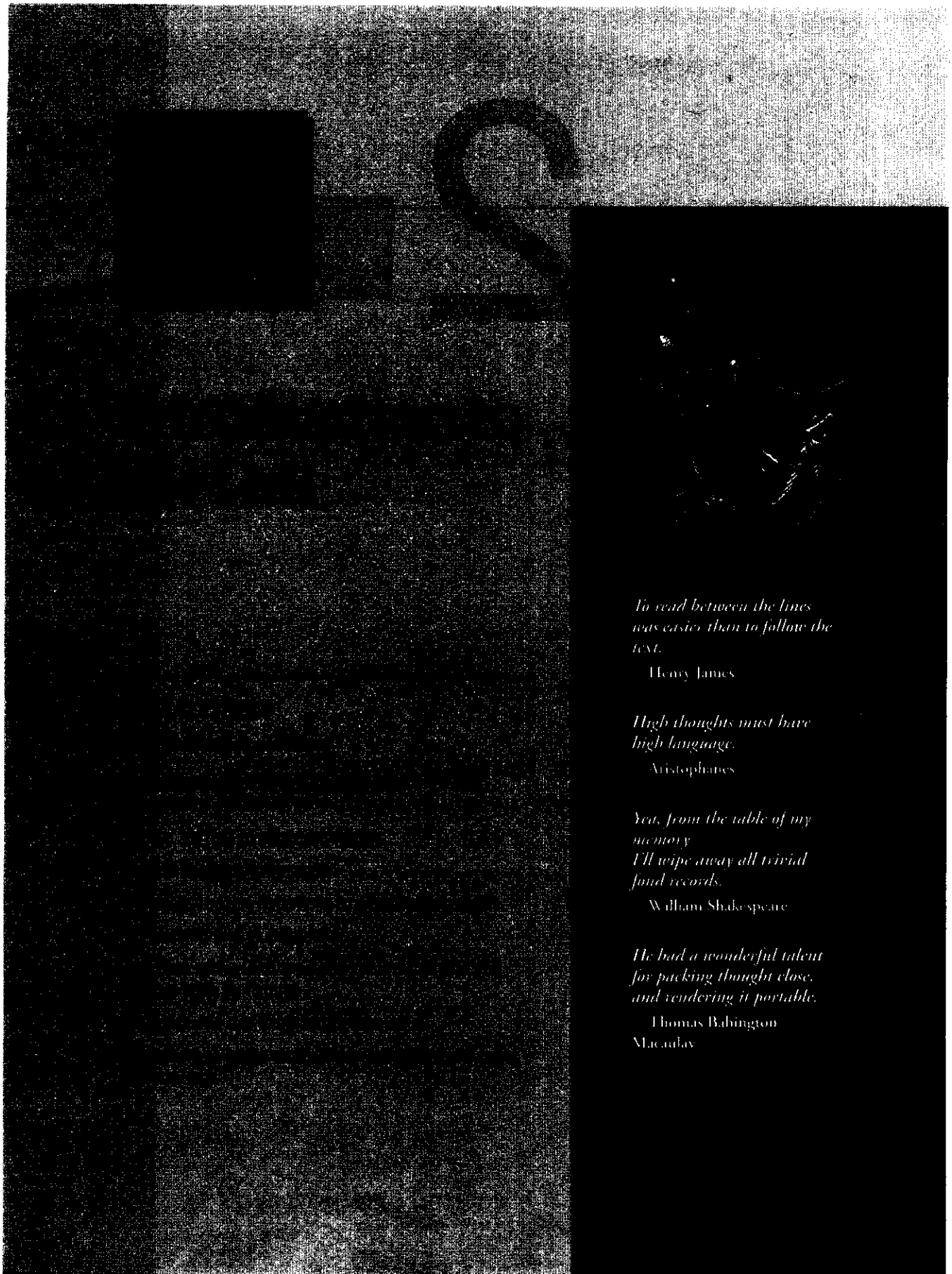
2

The Ajax Client

... the challenges are for the designers of these applications: to forget what we think we know about the limitations of the web, and begin to imagine a wider, richer range of possibilities. It's going to be fun.

—Jesse James Garrett



To read between the lines was easier than to follow the text.

Henry James

High thoughts must have high language.

Aristophanes

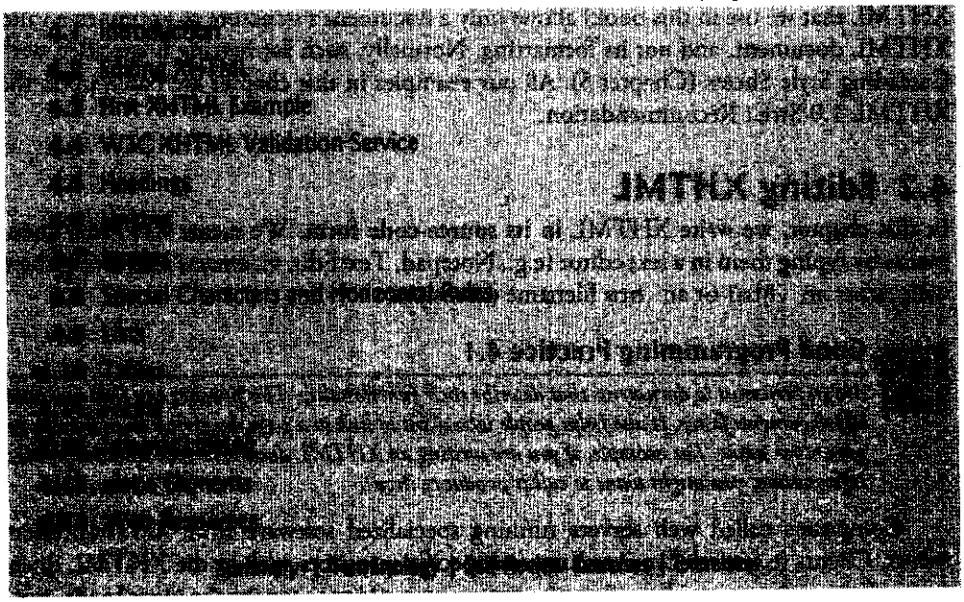
Yea, from the table of my memory
I'll wipe away all trivial fond records.

William Shakespeare

He had a wonderful talent for packing thought close, and rendering it portable.

Thomas Babington Macaulay

Outline



4.1 Introduction

Welcome to the world of opportunity created by the World Wide Web. The Internet is almost four decades old, but it wasn't until the web's growth in popularity in the 1990s and the recent start of the Web 2.0 era that the explosion of opportunity we are experiencing began. Exciting new developments occur almost daily—the pace of innovation is unprecedented. In this chapter, you'll develop your own web pages. As the book proceeds, you'll create increasingly appealing and powerful web pages. Later in the book, you'll learn how to create complete web-based applications with databases and user interfaces.

This chapter begins unlocking the power of web-based application development with XHTML—the **Extensible HyperText Markup Language**. Later in the chapter, we introduce more sophisticated XHTML techniques such as **internal linking** for easier page navigation, **forms** for collecting information from a web-page visitor and **tables**, which are particularly useful for structuring information from **databases** (i.e., software that stores structured sets of data). In the next chapter, we discuss a technology called **Cascading Style Sheets**™ (CSS), a technology that makes web pages more visually appealing.

Unlike procedural programming languages such as C, C++, or Java, XHTML is a **markup language** that specifies the format of the text that is displayed in a web browser such as Microsoft's Internet Explorer or Mozilla Firefox.

One key issue when using XHTML is the separation of the **presentation** of a document (i.e., the document's appearance when rendered by a browser) from the **structure** of the document's information. XHTML is based on HTML (HyperText Markup Language)—a legacy technology of the World Wide Web Consortium (W3C). In HTML, it was common to specify both the document's structure and its formatting. Formatting might specify where the browser placed an element in a web page or the fonts and colors used to display an element. The XHTML 1.0 Strict recommendation (the version of

XHTML that we use in this book) allows only a document's structure to appear in a valid XHTML document, and not its formatting. Normally, such formatting is specified with Cascading Style Sheets (Chapter 5). All our examples in this chapter are based upon the XHTML 1.0 Strict Recommendation.

4.2 Editing XHTML

In this chapter, we write XHTML in its **source-code form**. We create **XHTML documents** by typing them in a text editor (e.g., Notepad, TextEdit, vi, emacs) and saving them with either an `.html` or an `.htm` filename extension.

Good Programming Practice 4.1

Assign filenames to documents that describe their functionality. This practice can help you identify documents faster. It also helps people who want to link to a page, by giving them an easy-to-remember name. For example, if you are writing an XHTML document that contains product information, you might want to call it `products.html`.

Computers called **web servers** running specialized software store XHTML documents. Clients (e.g., web browsers) request specific **resources** such as the XHTML documents from web servers. For example, typing `www.deitel.com/books/downloads.html` into a web browser's address field requests `downloads.html` from the `books` directory on the web server running at `www.deitel.com`. We discuss web servers in detail in Chapter 21. For now, we simply place the XHTML documents on our computer and render them by opening them locally with a web browser such as Internet Explorer or Firefox.

4.3 First XHTML Example

This chapter presents XHTML markup and provides screen captures that show how a browser renders (i.e., displays) the XHTML. You can download the examples from `www.deitel.com/books/iw3htp4`. Every XHTML document we show has line numbers for the reader's convenience—these line numbers are not part of the XHTML documents. As you read this book, open each XHTML document in your web browser so you can view and interact with it as it was originally intended.

Figure 4.1 is an XHTML document named `main.html`. This first example displays the message "Welcome to XHTML!" in the browser. The key line in the program is line 13, which tells the browser to display "Welcome to XHTML!" Now let us consider each line of the program.

Lines 1–3 are required in XHTML documents to conform with proper XHTML syntax. For now, copy and paste these lines into each XHTML document you create. The meaning of these lines is discussed in detail in Chapter 14.

Lines 5–6 are **XHTML comments**. XHTML document creators insert comments to improve markup readability and describe the content of a document. Comments also help other people read and understand an XHTML document's markup and content. Comments do not cause the browser to perform any action when the user loads the XHTML document into the web browser to view it. XHTML comments always start with `<!--` and end with `-->`. Each of our XHTML examples includes comments that specify the figure number and filename and provide a brief description of the example's purpose. Subsequent examples include comments in the markup, especially to highlight new features.

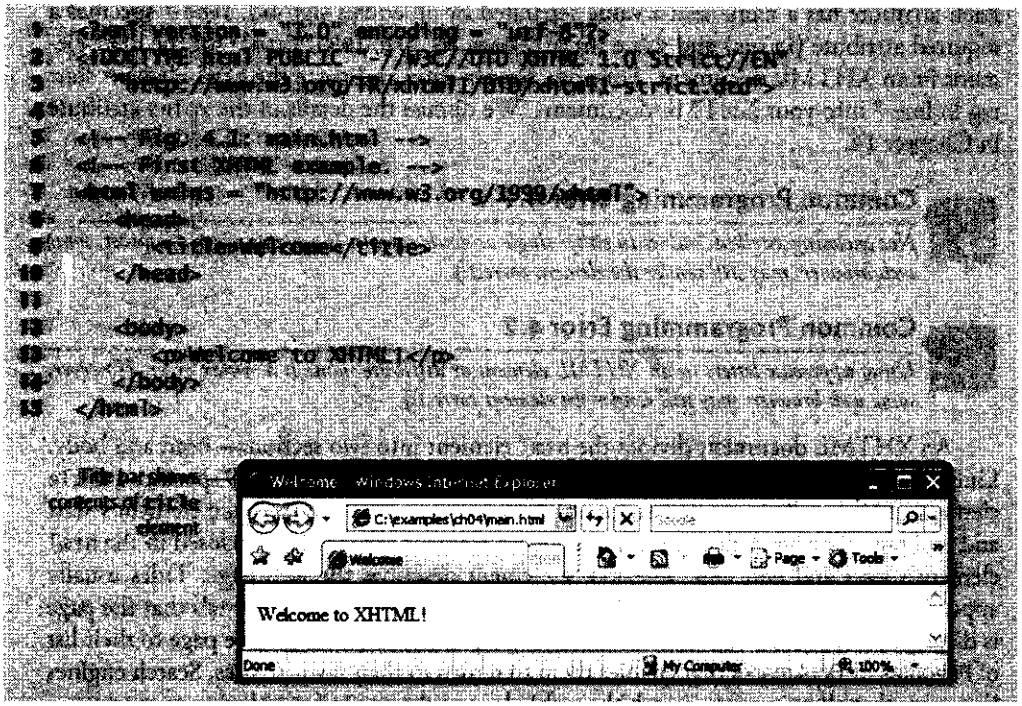


Fig. 4.1 | First XHTML example.

Good Programming Practice 4.2

Place comments throughout your markup. Comments help other programmers understand the markup, assist in debugging and list useful information that you do not want the browser to render. Comments also help you understand your own markup when you revisit a document to modify or update it in the future.

XHTML markup contains text that represents the content of a document and **elements** that specify a document's structure. Some important elements of an XHTML document are the `html` element, the **head** element and the **body** element. The `html` element encloses the **head section** (represented by the `head` element) and the **body section** (represented by the `body` element). The head section contains information about the XHTML document, such as its **title**. The head section also can contain special document formatting instructions called **style sheets** and client-side programs called **scripts** for creating dynamic web pages. (We introduce style sheets in Chapter 5 and scripting with JavaScript in Chapter 6.) The body section contains the page's content that the browser displays when the user visits the web page.

XHTML documents delimit an element with **start** and **end** tags. A start tag consists of the element name in angle brackets (e.g., `<html>`). An end tag consists of the element name preceded by a forward slash (/) in angle brackets (e.g., `</html>`). In this example, lines 7 and 15 define the start and end of the `html` element. Note that the end tag in line 15 has the same name as the start tag, but is preceded by a / inside the angle brackets. Many start tags have **attributes** that provide additional information about an element. Browsers can use this additional information to determine how to process the element.

Each attribute has a **name** and a **value** separated by an equals sign (=). Line 7 specifies a required attribute (`xmlns`) and value (`http://www.w3.org/1999/xhtml`) for the `html` element in an XHTML document. For now, simply copy and paste the `html` element start tag in line 7 into your XHTML documents. We discuss the details of the `xmlns` attribute in Chapter 14.

Common Programming Error 4.1

Not enclosing attribute values in either single or double quotes is a syntax error. However, some web browsers may still render the element correctly.

Common Programming Error 4.2

Using uppercase letters in an XHTML element or attribute name is a syntax error. However, some web browsers may still render the element correctly.

An XHTML document divides the `html` element into two sections—head and body. Lines 8–10 define the web page's head section with a `head` element. Line 9 specifies a `title` element. This is called a **nested element** because it is enclosed in the `head` element's start and end tags. The `head` element is also a nested element because it is enclosed in the `html` element's start and end tags. The `title` element describes the web page. Titles usually appear in the **title bar** at the top of the browser window, in the browser tab that the page is displayed on, and also as the text identifying a page when users add the page to their list of **Favorites** or **Bookmarks** that enables them to return to their favorite sites. Search engines (i.e., sites that allow users to search the web) also use the `title` for indexing purposes.

Good Programming Practice 4.3

Indenting nested elements emphasizes a document's structure and promotes readability.

Common Programming Error 4.3

XHTML does not permit tags to overlap—a nested element's end tag must appear in the document before the enclosing element's end tag. For example, the nested XHTML tags `<head><title>hello</head></title>` *cause a syntax error, because the enclosing* `head` *element's ending* `</head>` *tag appears before the nested* `title` *element's ending* `</title>` *tag.*

Good Programming Practice 4.4

Use a consistent `title`*-naming convention for all pages on a site. For example, if a site is named "Bailey's Website," then the* `title` *of the contact page might be "Bailey's Website - Contact." This practice can help users better understand the website's structure.*

Line 12 begins the document's `body` element. The body section of an XHTML document specifies the document's content, which may include text and elements.

Some elements, such as the **paragraph element** (denoted with `<p>` and `</p>`) in line 13, mark up text for display in a browser. All the text placed between the `<p>` and `</p>` tags forms one paragraph. When the browser renders a paragraph, a blank line usually precedes and follows paragraph text.

This document ends with two end tags (lines 14–15). These tags close the `body` and `html` elements, respectively. The `</html>` tag in an XHTML document informs the browser that the XHTML markup is complete.

To open an XHTML example from this chapter, open the folder where you saved the book's examples, browse to the ch04 folder and double click the file to open it in your default web browser. At this point your browser window should appear similar to the sample screen capture shown in Fig. 4.1. (Note that we resized the browser window to save space in the book.)

4.4 W3C XHTML Validation Service

Programming web-based applications can be complex, and XHTML documents must be written correctly to ensure that browsers process them properly. To promote correctly written documents, the World Wide Web Consortium (W3C) provides a **validation service** (`validator.w3.org`) for checking a document's syntax. Documents can be validated by providing a URL that specifies the file's location, by uploading a file to `validator.w3.org/file-upload.html` or by pasting code directly into a text area. Uploading a file copies the file from the user's computer to another computer on the Internet. The W3C's web page indicates that the service name is **MarkUp Validation Service** and that the validation service is able to validate the syntax of XHTML documents. All the XHTML examples in this book have been validated successfully using `validator.w3.org`.

By clicking **Choose...**, users can select files on their own computers for upload. After selecting a file, clicking the **Check** button uploads and validates the file. If a document contains syntax errors, the validation service displays error messages describing the errors.

Error-Prevention Tip 4.1

Most current browsers attempt to render XHTML documents even if they are invalid. This often leads to unexpected and possibly undesirable results. Use a validation service, such as the W3C **MarkUp Validation Service**, *to confirm that an XHTML document is syntactically correct.*

4.5 Headings

Some text in an XHTML document may be more important than other text. For example, the text in this section is considered more important than a footnote. XHTML provides six **headings**, called **heading elements**, for specifying the relative importance of information. Figure 4.2 demonstrates these elements (`h1` through `h6`). Heading element `h1` (line 13) is considered the most significant heading and is typically rendered in a larger font than the other five headings (lines 14–18). Each successive heading element (i.e., `h2`, `h3`, etc.) is typically rendered in a progressively smaller font.

Portability Tip 4.1

The text size used to display each heading element can vary significantly between browsers. In Chapter 5, we discuss how to control the text size and other text properties.

Look-and-Feel Observation 4.1

Placing a heading at the top of every XHTML page helps viewers understand the purpose of each page.

Look-and-Feel Observation 4.2

Use larger headings to emphasize more important sections of a web page.

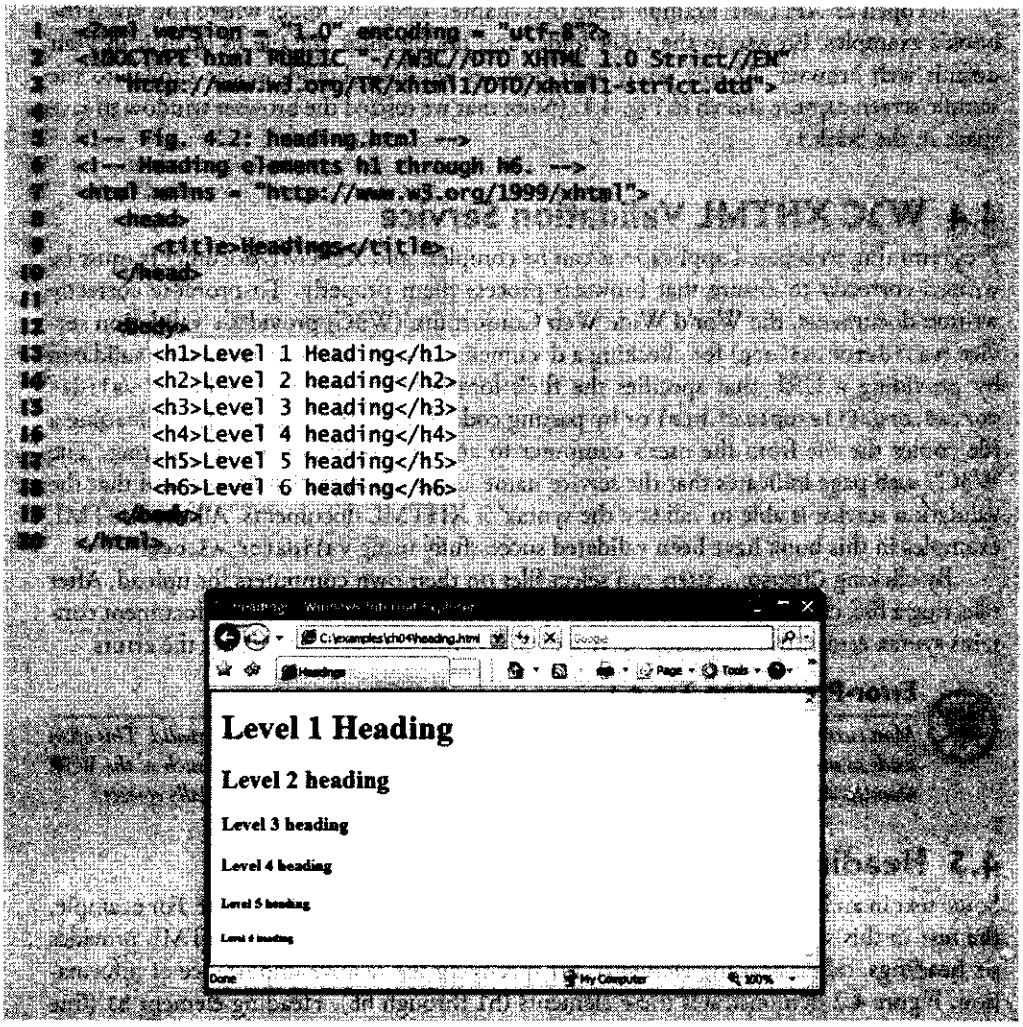


Fig. 4.2 | Heading elements `h1` through `h6`.

4.6 Linking

One of the most important XHTML features is the **hyperlink**, which references (or **links** to) other resources, such as XHTML documents and images. When a user clicks a hyperlink, the browser tries to execute an action associated with it (e.g., navigate to a URL, open an e-mail client, etc.). In XHTML, both text and images can act as hyperlinks. Web browsers typically underline text hyperlinks and color their text blue by default, so that users can distinguish hyperlinks from plain text. In Fig. 4.3, we create text hyperlinks to four different websites.

Line 14 introduces the **`strong` element**, which indicates that its contents has high importance. Browsers typically display such text in a bold font.

Links are created using the **`a` (anchor)** element. Line 17 defines a hyperlink to the URL assigned to attribute **`href`**, which specifies the location of a linked resource, such as

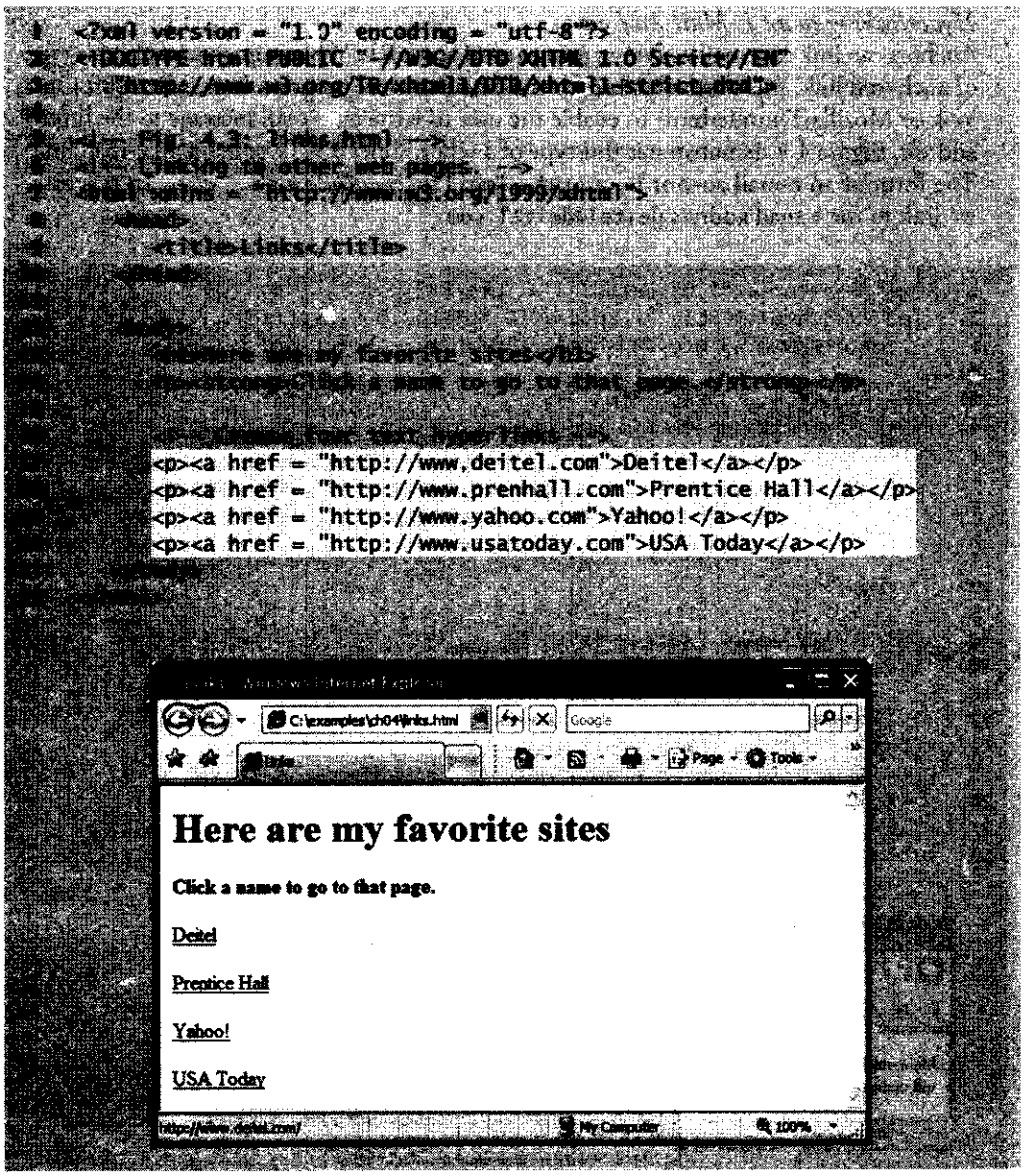


Fig. 4.3 | Linking to other web pages.

a web page, a file or an e-mail address. This particular anchor element links the text `Deitel` to a web page located at `http://www.deitel.com`. When a URL does not indicate a specific document on the website, the web server returns a default web page. This page is often called `index.html`; however, most web servers can be configured to use any file as the default web page for the site. If the web server cannot locate a requested document, it returns an error indication to the web browser, and the browser displays a web page containing an error message to the user.

Hyperlinking to an E-Mail Address

Anchors can link to e-mail addresses using a `mailto:` URL. When someone clicks this type of anchored link, most browsers launch the default e-mail program (e.g., Microsoft Outlook or Mozilla Thunderbird) to enable the user to write an e-mail message to the linked address. Figure 4.4 demonstrates this type of anchor. Lines 15–17 contain an e-mail link. The form of an e-mail anchor is `<a href = "mailto:`*emailAddress*`">...</a>`. In this case, we link to the e-mail address `deitel@deitel.com`.

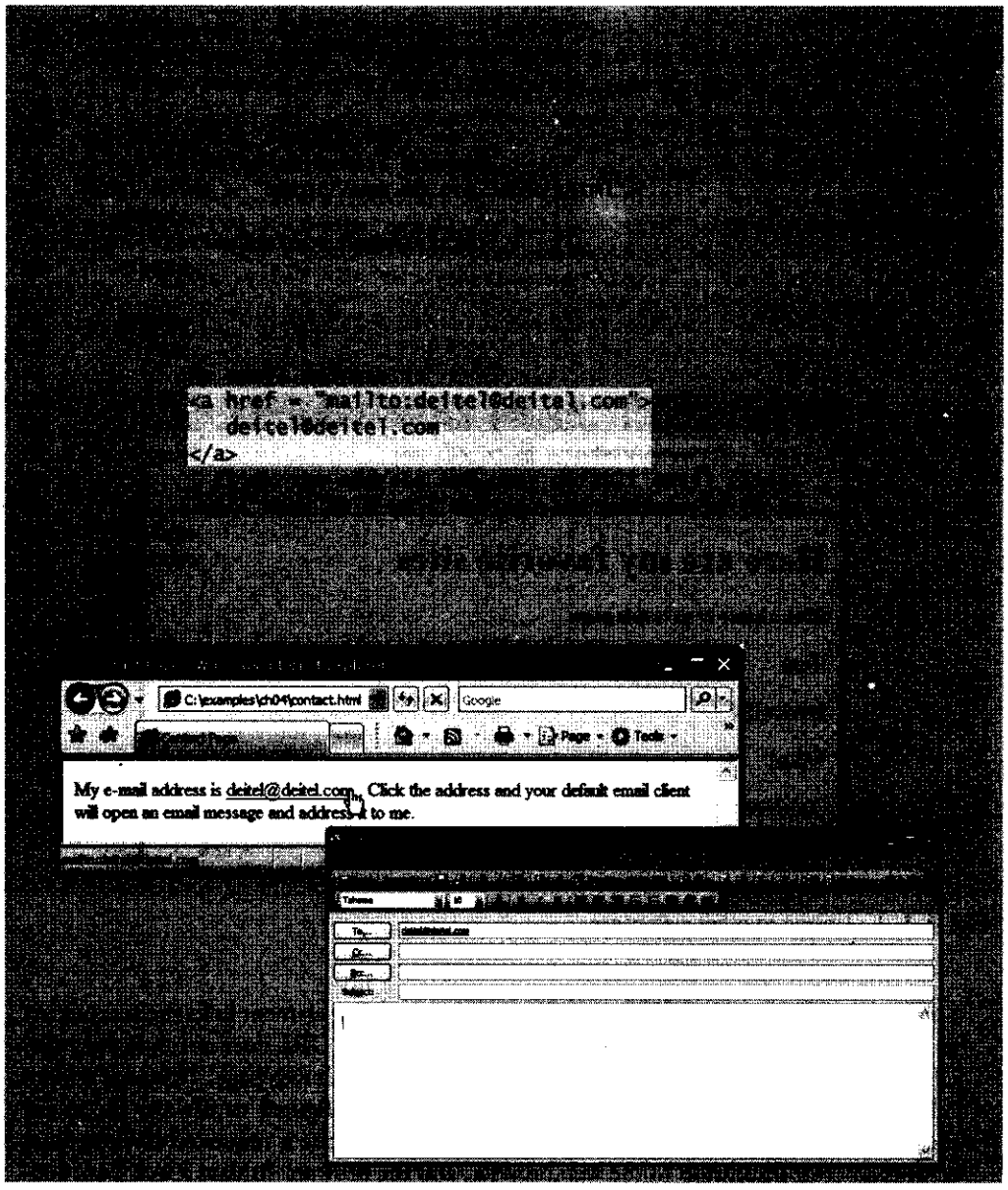


Fig. 4.4 | Linking to an e-mail address.

4.7 Images

The examples discussed so far demonstrate how to mark up documents that contain only text. However, most web pages contain both text and images. In fact, images are an equally important, if not essential, part of web-page design. The three most popular image formats used by web developers are Graphics Interchange Format (GIF), Joint Photographic Experts Group (JPEG) and Portable Network Graphics (PNG) images. Users can create images using specialized software, such as Adobe Photoshop Elements (`www.adobe.com`), G.I.M.P. (`http://www.gimp.org`) and Inkscape (`http://www.inkscape.org`). Images may also be acquired from various websites. Figure 4.5 demonstrates how to incorporate images into web pages.

Lines 14–15 use an **`img` element** to insert an image in the document. The image file's location is specified with the `img` element's **`src` attribute**. This image is located in the same directory as the XHTML document, so only the image's filename is required. Optional attributes **`width`** and **`height`** specify the image's width and height, respectively. You can scale an image by increasing or decreasing the values of the image `width` and `height` attributes. If these attributes are omitted, the browser uses the image's actual width and height. Images are measured in **pixels** ("picture elements"), which represent dots of color on the screen. Any image-editing program will have a feature that displays the dimensions, in pixels, of an image. The image in Fig. 4.5 is 92 pixels wide and 120 pixels high.

Good Programming Practice 4.5

Always include the `width` and the `height` of an image inside the `<img>` tag. When the browser loads the XHTML file, it will know immediately from these attributes how much screen space to provide for the image and will lay out the page properly, even before it downloads the image.

Performance Tip 4.1

Including the `width` and `height` attributes in an `<img>` tag can result in the browser's loading and rendering pages faster.

Common Programming Error 4.4

Entering new dimensions for an image that change its inherent width-to-height ratio distorts the appearance of the image. For example, if your image is 200 pixels wide and 100 pixels high, you should ensure that any new dimensions have a 2:1 width-to-height ratio.

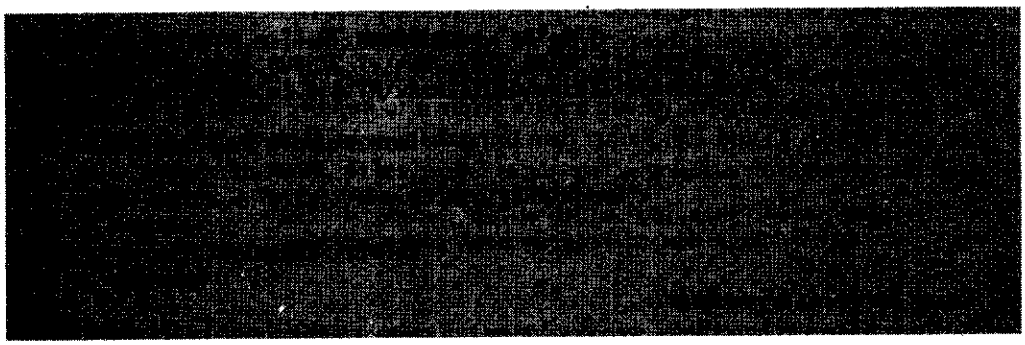


Fig. 4.5 | Images in XHTML files. (Part 1 of 2.)

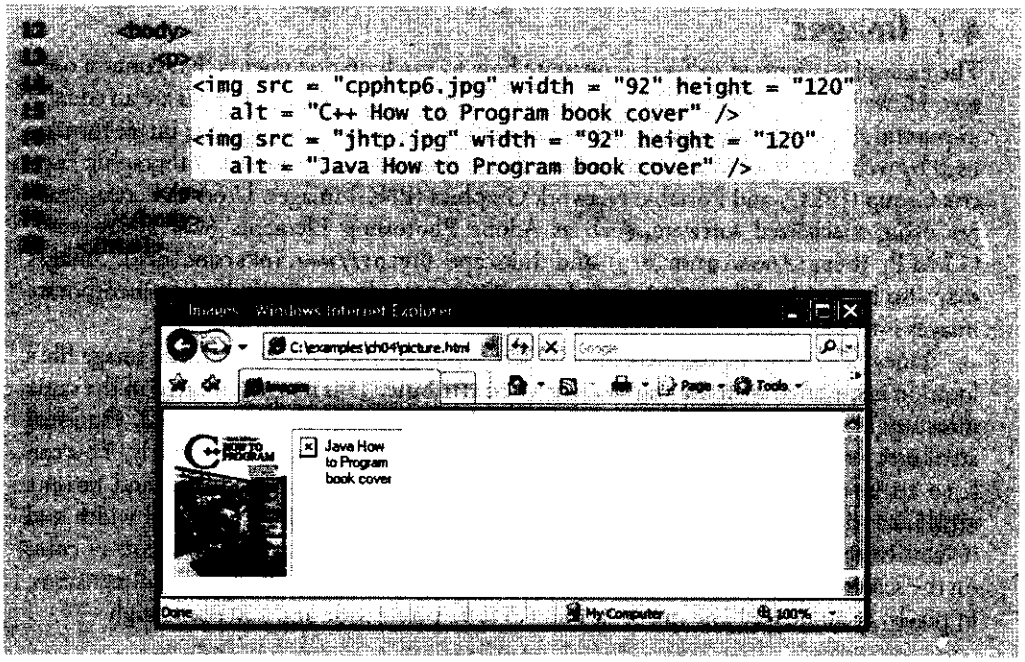


Fig. 4.5 | Images in XHTML files. (Part 2 of 2.)

Every `img` element in an XHTML document must have an **`alt` attribute**. If a browser cannot render an image, the browser displays the `alt` attribute's value. A browser may not be able to render an image for several reasons. It may not support images—as is the case with a **text-based browser** (i.e., a browser that can display only text)—or the client may have disabled image viewing to reduce download time. Figure 4.5 shows Internet Explorer 7 rendering a red x symbol and displaying the `alt` attribute's value, signifying that the image (`jhtp.jpg`) cannot be found.

The `alt` attribute helps you create **accessible** web pages for users with disabilities, especially those with vision impairments who use text-based browsers. Specialized software called a **speech synthesizer** can "speak" the `alt` attribute's value so that a user with a visual impairment knows what the browser is displaying.

Some XHTML elements (called **empty elements**) contain only attributes and do not mark up text (i.e., text is not placed between the start and end tags). Empty elements (e.g., `img`) must be terminated, either by using the **forward slash character** (/) inside the closing right angle bracket (>) of the start tag or by explicitly including the end tag. When using the forward slash character, we add a space before it to improve readability (as shown at the ends of lines 15 and 17). Rather than using the forward slash character, lines 16–17 could be written with a closing `</img>` tag as follows:

```
<img src = "jhtp.jpg" width = "92" height = "120"
   alt = "Java How to Program book cover"></img>
```

Using Images as Hyperlinks

By using images as hyperlinks, web developers can create graphical web pages that link to other resources. In Fig. 4.6, we create six different image hyperlinks.

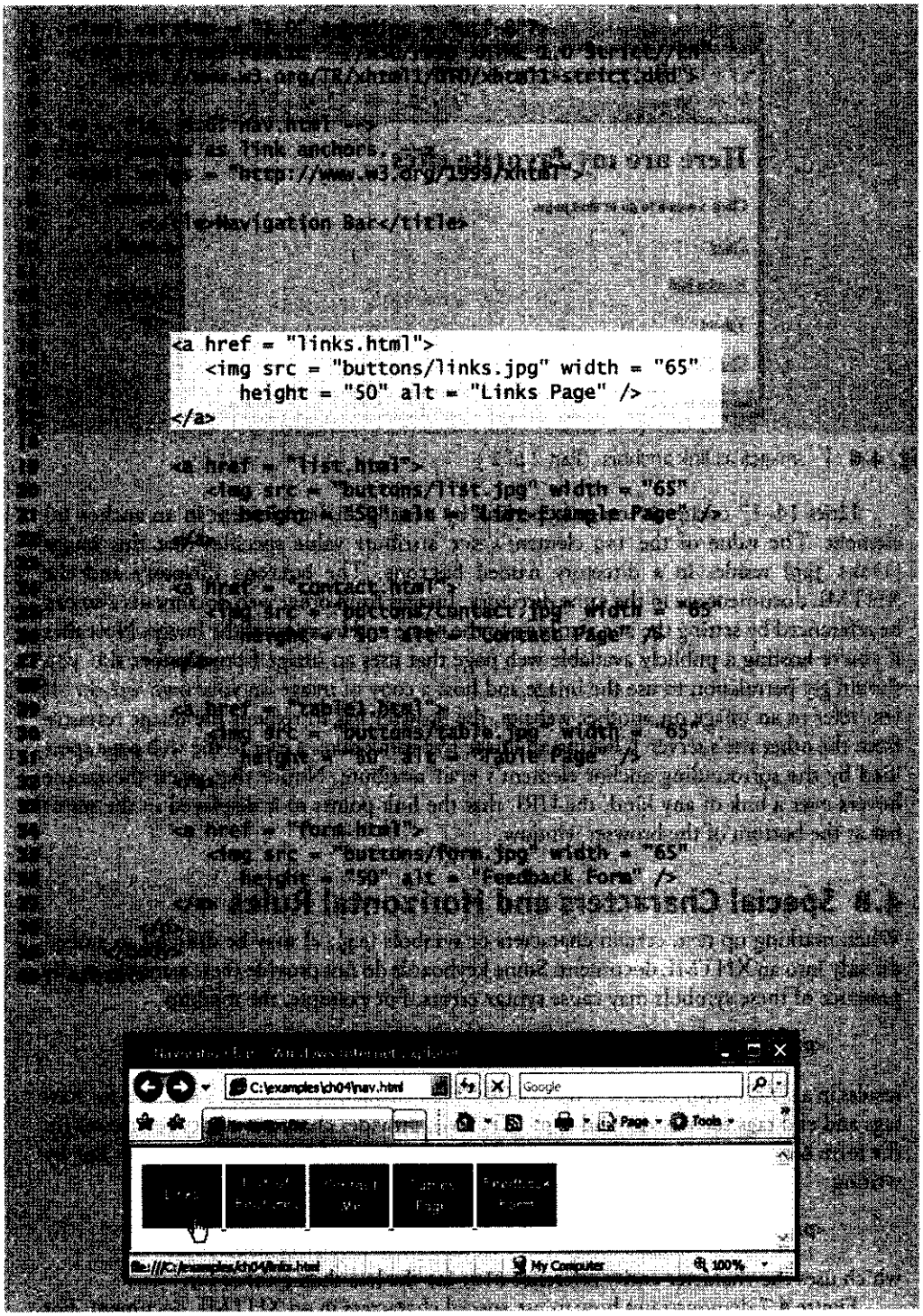


Fig. 4.6 | Images as link anchors. (Part 1 of 2.)

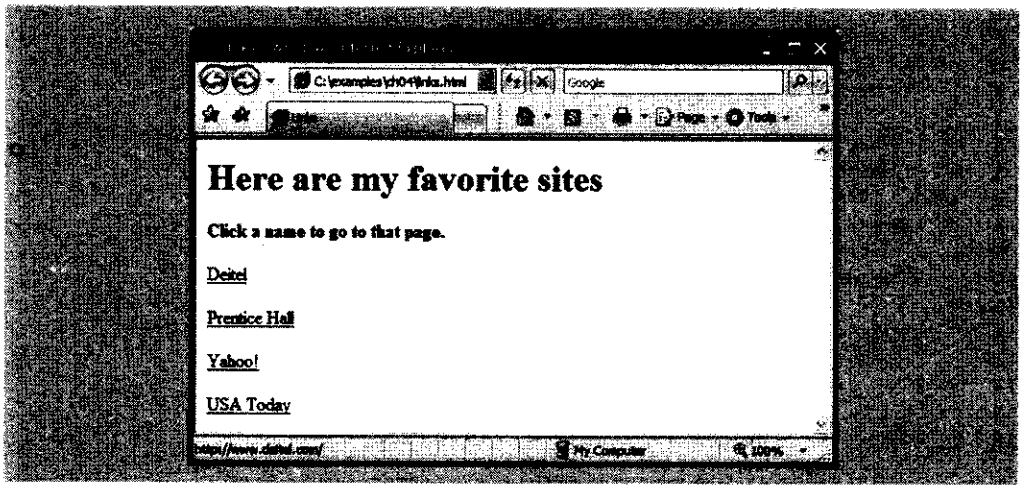


Fig. 4.6 | Images as link anchors. (Part 2 of 2.)

Lines 14–17 create an **image hyperlink** by nesting an `img` element in an anchor (`a`) element. The value of the `img` element's `src` attribute value specifies that this image (`links.jpg`) resides in a directory named `buttons`. The `buttons` directory and the XHTML document are in the same directory. Images from other web documents also can be referenced by setting the `src` attribute to the name and location of the image. Note that if you're hosting a publicly available web page that uses an image from another site, you should get permission to use the image and host a copy of image on your own website. If you refer to an image on another website, the browser has to request the image resource from the other site's server. Clicking an image hyperlink takes a user to the web page specified by the surrounding anchor element's `href` attribute. Notice that when the mouse hovers over a link of any kind, the URL that the link points to is displayed in the status bar at the bottom of the browser window.

4.8 Special Characters and Horizontal Rules

When marking up text, certain characters or symbols (e.g., <) may be difficult to embed directly into an XHTML document. Some keyboards do not provide these symbols, or the presence of these symbols may cause syntax errors. For example, the markup

```
<p>if x < 10 then increment x by 1</p>
```

results in a syntax error because it uses the less-than character (<), which is reserved for start tags and end tags such as and . XHTML provides **character entity references** (in the form *&code;*) for representing special characters. We could correct the previous line by writing

```
<p>if x &lt; 10 then increment x by 1</p>
```

which uses the character entity reference **`<`** for the less-than symbol (<).

Figure 4.7 demonstrates how to use special characters in an XHTML document. For a list of special characters, see Appendix A, XHTML Special Characters.

Lines 24–25 contain other special characters, which can be expressed as either character entity references (coded using word abbreviations such as `&amp;` for ampersand and `&copy;` for copyright) or **numeric character references**—decimal or **hexadecimal** (**hex**) values representing special characters. For example, the & character is represented in decimal and hexadecimal notation as `&#38;` and `&#x26;`, respectively. Hexadecimal numbers are base 16 numbers—digits in a hexadecimal number have values from 0 to 15 (a total of 16 different values). The letters A–F represent the hexadecimal digits corresponding to decimal values 10–15. Thus in hexadecimal notation we can have numbers like 876 consisting solely of decimal-like digits, numbers like DA19F consisting of digits and letters, and numbers like DCB consisting solely of letters.

```
<hr /> <!-- inserts a horizontal rule -->

<p>All information on this site is <strong>&amp;copy;
   Deitel &amp;amp; Associates, Inc. 2007.</strong></p>

<p><del>You may download 3.14 x 10<sup>2</sup>
   characters worth of information from this site.</del>
   Only <sub>one</sub> download per hour is permitted.</p>
<p><em>Note: &amp;lt; &amp;frac14; of the information
   presented here is updated daily.</em></p>
```

Fig. 4.7 | Inserting special characters. (Part 1 of 2.)

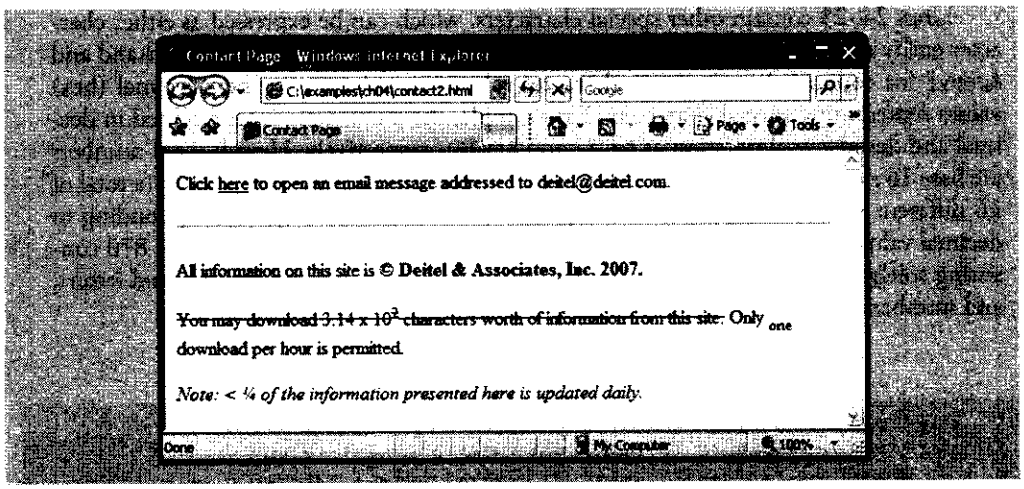


Fig. 4.7 | Inserting special characters. (Part 2 of 2.)

In lines 31–33, we introduce four new elements. Most browsers render the **del** element as strike-through text. With this format users can easily indicate document revisions. To **superscript** text (i.e., raise text above the baseline and decreased font size) or **subscript** text (i.e., lower text below the baseline and decreased font size), use the **sup** or **sub** element, respectively. The paragraph in lines 34–35 contains an **em element**, which indicates that its contents should be emphasized. Browsers usually render `em` elements in an italic font. We also use character entity reference `<` for a less-than sign and `&frac14;` for the fraction 1/4 (line 34).

In addition to special characters, this document introduces a **horizontal rule**, indicated by the **<hr />** tag in line 22. Most browsers render a horizontal rule as a horizontal line with a blank line above and below it.

4.9 Lists

Up to this point, we have presented basic XHTML elements and attributes for linking to resources, creating headings, using special characters and incorporating images. In this section, we discuss how to organize information on a web page using lists. In the next section, we introduce another feature for organizing information, called a table. Figure 4.8 displays text in an **unordered list** (i.e., a list that does not order its items by letter or number). The unordered list element **ul** creates a list in which each item begins with a bullet symbol (called a disc). Each entry in an unordered list (element `ul` in line 17) is an **li (list item)** element (lines 19–22). Most web browsers render each `li` element on a new line with a bullet symbol indented from the beginning of the line.

Fig. 4.8 | Unordered list containing hyperlinks. (Part 1 of 2.)

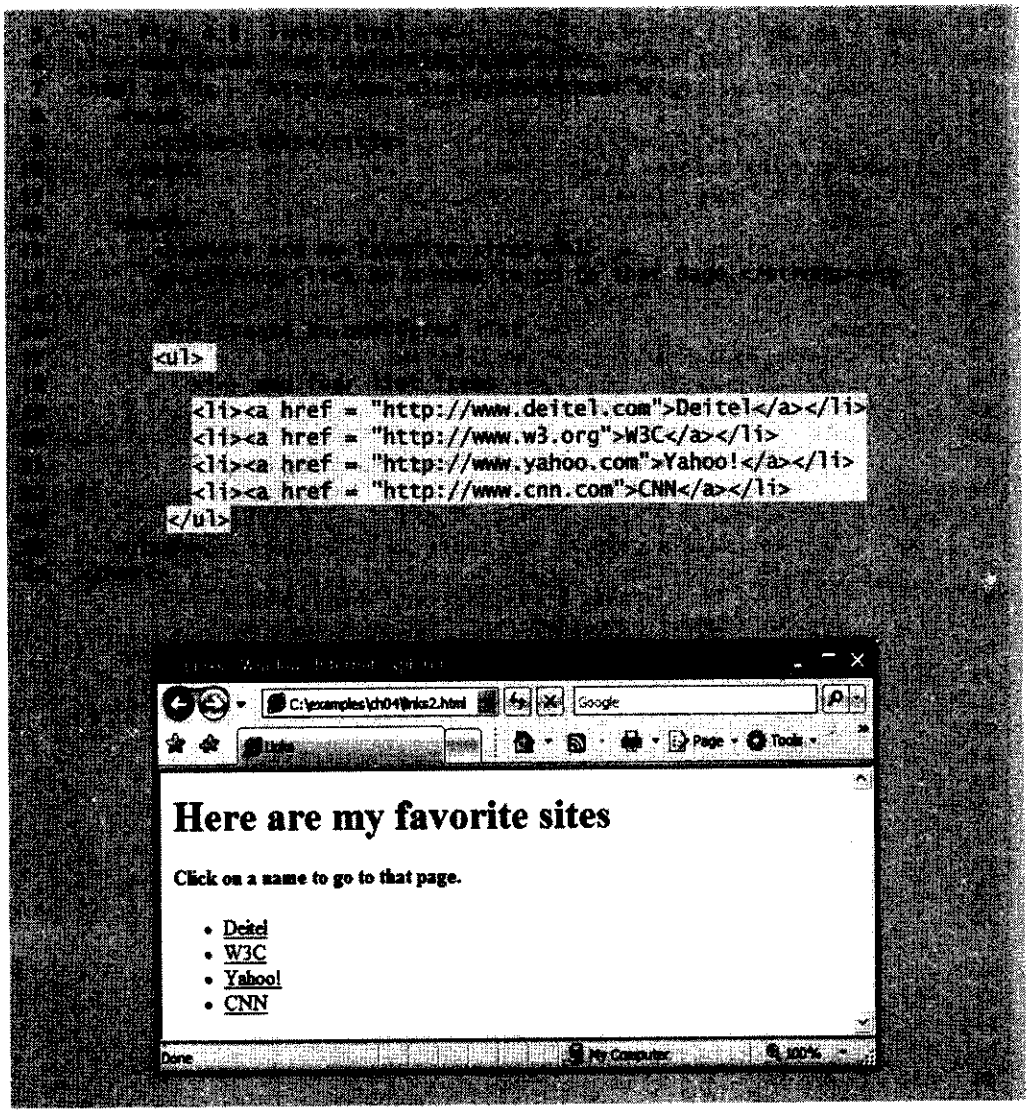
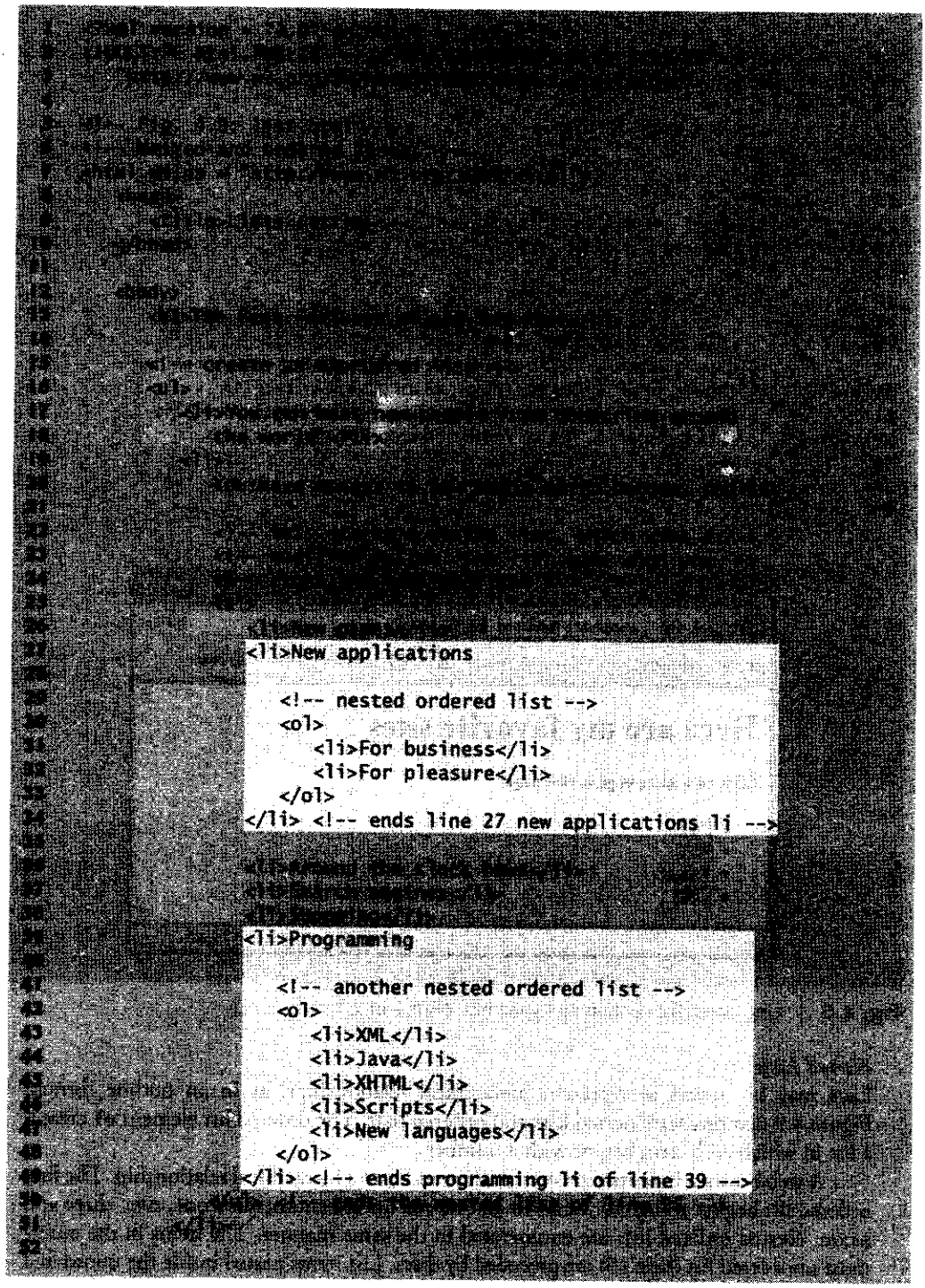


Fig. 4.8 | Unordered list containing hyperlinks. (Part 2 of 2.)

Nested Lists

Lists may be nested to represent hierarchical relationships, as in an outline format. Figure 4.9 demonstrates nested lists and **ordered lists**. The ordered list element `ol` creates a list in which each item begins with a number.

A web browser indents each nested list to indicate a hierarchical relationship. The first ordered list begins at line 30. Items in an ordered list are enumerated one, two, three and so on. Nested ordered lists are enumerated in the same manner. The items in the outermost unordered list (line 16) are preceded by discs. List items nested inside the unordered list of line 16 are preceded by circular bullets. Although not demonstrated in this example, subsequent nested list items are preceded by square bullets.



```
<li>New applications

   <!-- nested ordered list -->
   <ol>
      <li>For business</li>
      <li>For pleasure</li>
   </ol>
</li> <!-- ends line 27 new applications li -->
```

```
<li>Programming

   <!-- another nested ordered list -->
   <ol>
      <li>XML</li>
      <li>Java</li>
      <li>XHTML</li>
      <li>Scripts</li>
      <li>New languages</li>
   </ol>
</li> <!-- ends programming li of line 39 -->
```

Fig. 4.9 | Nested and ordered lists. (Part 1 of 2.)

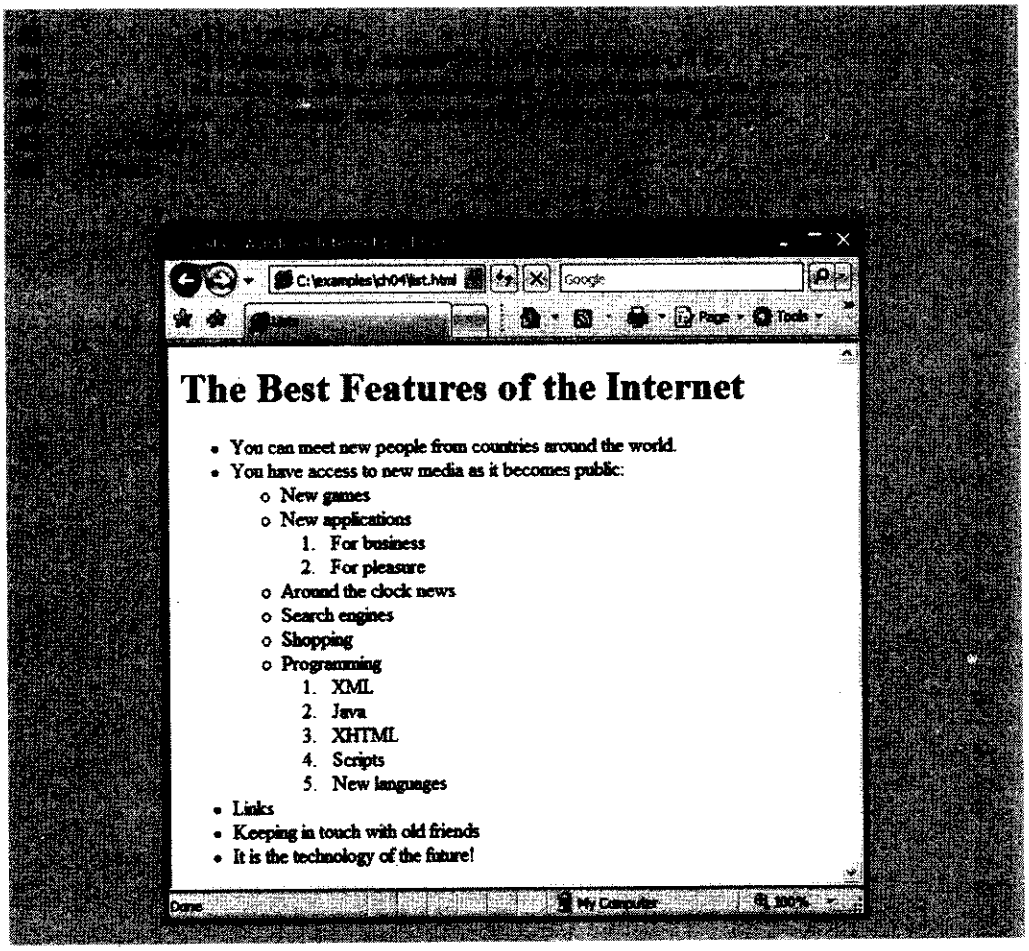


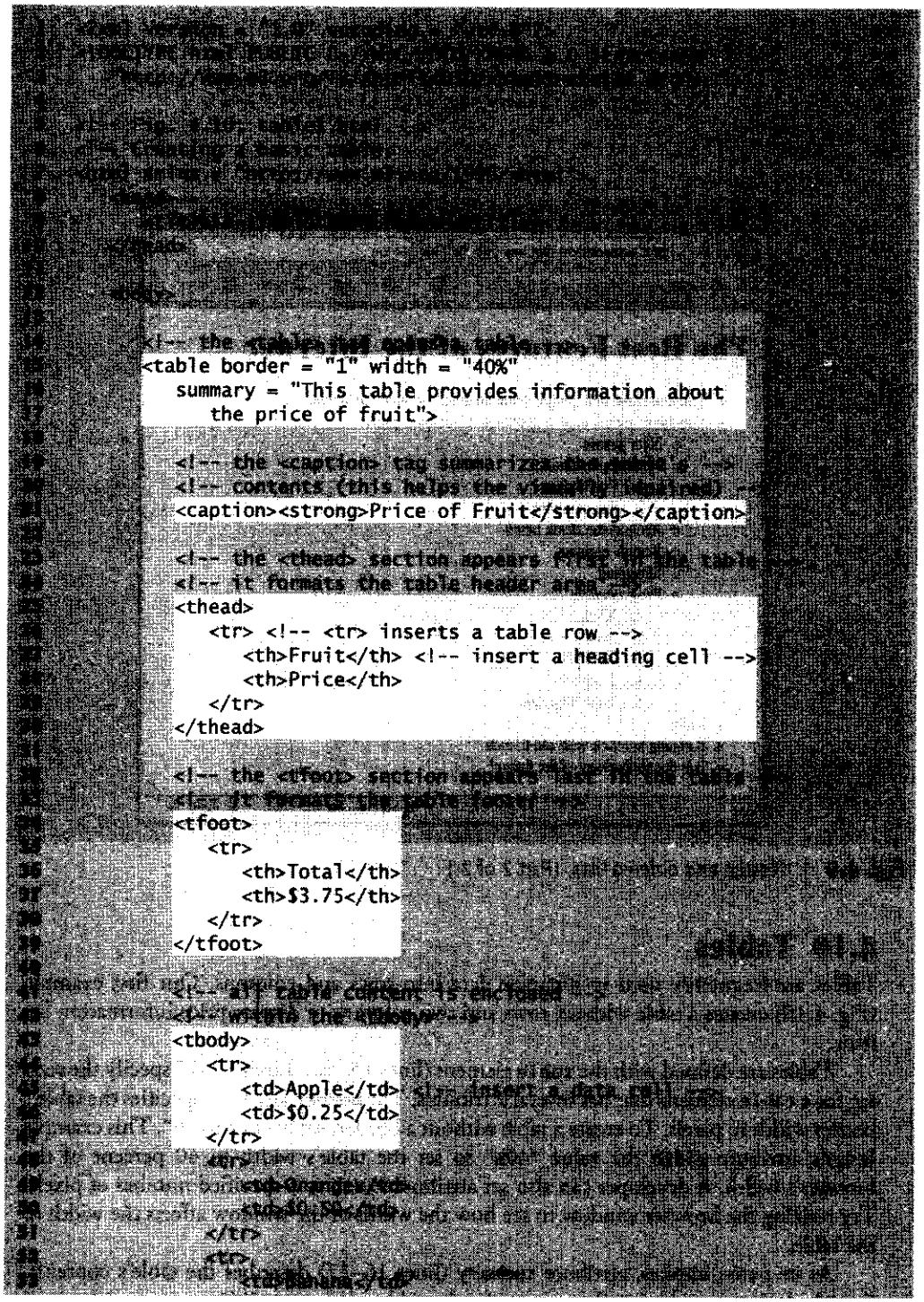
Fig. 4.9 | Nested and ordered lists. (Part 2 of 2.)

4.10 Tables

Tables are frequently used to organize data into rows and columns. Our first example (Fig. 4.10) creates a table with six rows and two columns to display price information for fruit.

Tables are defined with the **`table`** element (lines 15–62). Lines 15–17 specify the start tag for a `table` element that has several attributes. The **`border`** attribute specifies the table's border width in pixels. To create a table without a border, set `border` to `"0"`. This example assigns attribute **`width`** the value `"40%"` to set the table's width to 40 percent of the browser's width. A developer can also set attribute `width` to a specified number of pixels. Try resizing the browser window to see how the width of the window affects the width of the table.

As its name implies, attribute **`summary`** (lines 16–17) describes the table's contents. Speech devices use this attribute to make the table more accessible to users with visual impairments. The **`caption`** element (line 21) describes the table's content and helps text-



```
<table border = "1" width = "40%"
   summary = "This table provides information about
      the price of fruit">

   <caption><strong>Price of Fruit</strong></caption>

   <thead>
      <tr> <!-- <tr> inserts a table row -->
         <th>Fruit</th> <!-- insert a heading cell -->
         <th>Price</th>
      </tr>
   </thead>

   <tfoot>
      <tr>
         <th>Total</th>
         <th>$3.75</th>
      </tr>
   </tfoot>

   <tbody>
      <tr>
         <td>Apple</td>
         <td>$0.25</td>
      </tr>
```

Fig. 4.10 | Creating a basic table. (Part 1 of 2.)

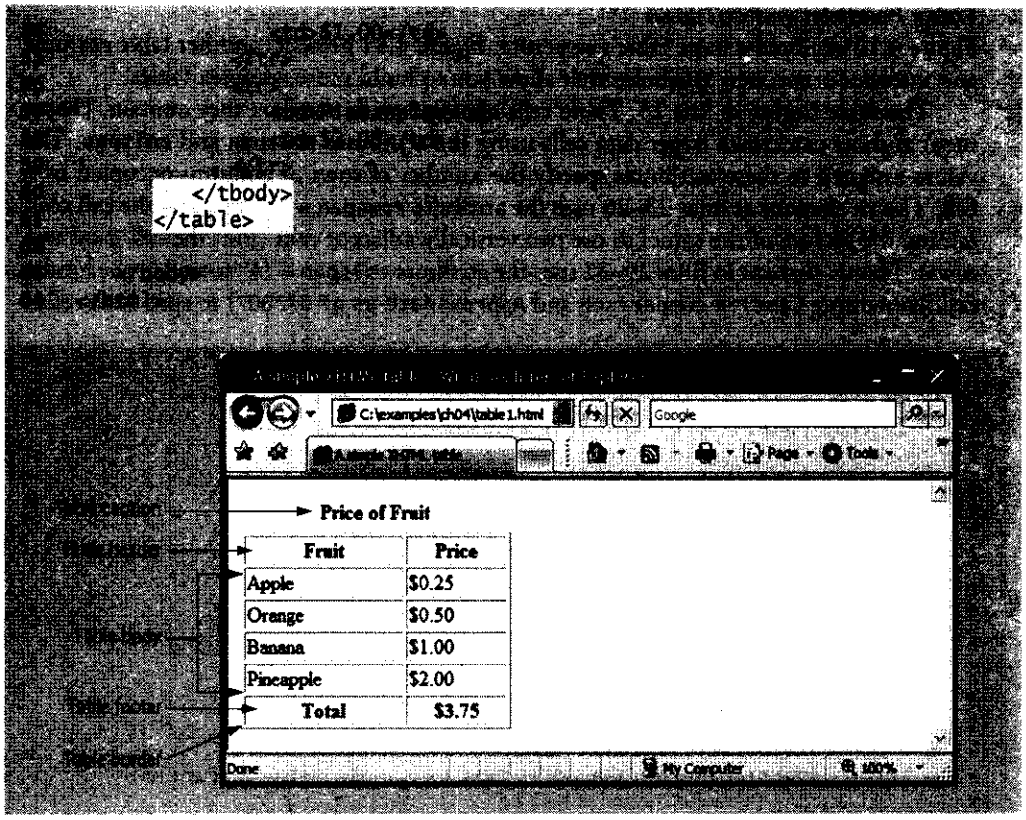


Fig. 4.10 | Creating a basic table. (Part 2 of 2.)

based browsers interpret the table data. Text inside the `<caption>` tag is rendered above the table by most browsers. Attribute `summary` and element `caption` are two of the many XHTML features that make web pages more accessible to users with disabilities.

A table has three distinct sections—**head**, **body** and **foot**. The head section (or header cell) is defined with a **thead** element (lines 25–30), which contains header information such as column names. Each **tr** element (lines 26–29) defines an individual **table row**. The columns in the head section are defined with **th** elements. Most browsers center text formatted by `th` (table header column) elements and display them in bold. Table header elements are nested inside table row elements (lines 27–28).

The body section, or **table body**, contains the table's primary data. The table body (lines 43–60) is defined in a **tbody** element. In the body, each `tr` element specifies one row. **Data cells** contain individual pieces of data and are defined with **td** (**table data**) elements in each row.

The foot section (lines 34–39) is defined with a **tfoot** (table foot) element. The text placed in the footer commonly includes calculation results and footnotes. Like other sections, the foot section can contain table rows, and each row can contain cells. As in the head section, cells in the foot section are created using `th` elements, instead of the `td` elements used in the table body. Note that the table foot section must be above the body section in the code, but the table foot displays at the bottom of the table.

Using rowspan and colspan

Figure 4.10 explored a basic table's structure. Figure 4.11 presents another table example and introduces two new attributes that allow you to build more complex tables.

The table begins in line 15. Table cells are sized to fit the data they contain. Document authors can create larger data cells using the attributes **rowspan** and **colspan**. The values assigned to these attributes specify the number of rows or columns occupied by a cell. The `th` element at lines 23–26 uses the attribute `rowspan = "2"` to allow the cell containing the picture of the camel to use two vertically adjacent cells (thus the cell *spans* two rows). The `th` element in lines 29–32 uses the attribute `colspan = "4"` to widen the header cell (containing `Camelid comparison` and `Approximate as of 6/2007`) to span four cells.

```
<!-- merge two rows -->
<th rowspan = "2">
   <img src = "camel.gif" width = "205"
      height = "167" alt = "Picture of a camel" />
</th>

<!-- merge four columns -->
<th colspan = "4">
   <h1>Camelid comparison</h1>
   <p>Approximate as of 6/2007</p>
</th>
```

Fig. 4.11 | Complex XHTML table. (Part 1 of 2.)

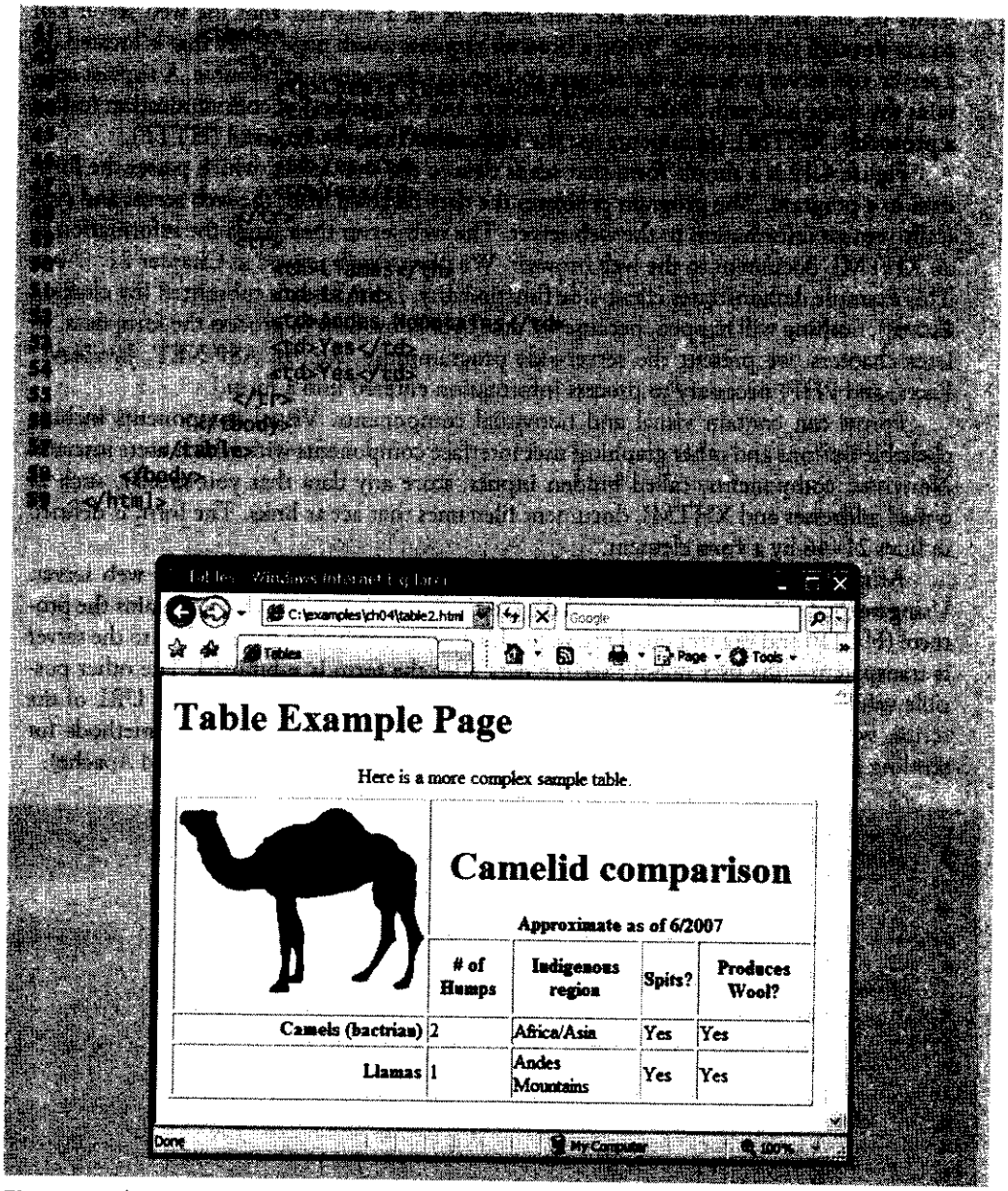


Fig. 4.11 | Complex XHTML table. (Part 2 of 2.)

4.11 Forms

When browsing websites, users often need to provide such information as search keywords, e-mail addresses and zip codes. XHTML provides a mechanism, called a form, for collecting data from a user.

Data that users enter on a web page is normally sent to a web server that provides access to a site's resources (e.g., XHTML documents, images). These resources are located

either on the same machine as the web server or on a machine that the web server can access through the network. When a browser requests a web page or file that is located on a server, the server processes the request and returns the requested resource. A request contains the name and path of the desired resource and the method of communication (called a **protocol**). XHTML documents use the Hypertext Transfer Protocol (HTTP).

Figure 4.12 is a simple form that sends data to the web server, which passes the form data to a program. The program processes the data received from the web server and typically returns information to the web server. The web server then sends the information as an XHTML document to the web browser. We discuss web servers in Chapter 21. [*Note:* This example demonstrates client-side functionality. If the form is submitted (by clicking **Submit**), nothing will happen, because we don't yet know how to process the form data. In later chapters, we present the server-side programming (e.g., in ASP.NET, JavaServer Faces, and PHP) necessary to process information entered into a form.]

Forms can contain visual and nonvisual components. Visual components include clickable buttons and other graphical user interface components with which users interact. Nonvisual components, called **hidden inputs**, store any data that you specify, such as e-mail addresses and XHTML document filenames that act as links. The form is defined in lines 21–46 by a **form** element.

Attribute **method** (line 21) specifies how the form's data is sent to the web server. Using `method = "post"` appends form data to the browser request, which contains the protocol (HTTP) and the requested resource's URL. This method of passing data to the server is transparent—the user doesn't see the data after the form is submitted. The other possible value, `method = "get"`, appends the form data directly to the end of the URL of the script, where it is visible in the browser's **Address** field. The *post* and *get* methods for sending form data are discussed in detail in Chapter 21, Web Servers (IIS and Apache).

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 4.12: form.html -->
<!-- Form with hidden fields and a text box. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Forms</title>
   </head>

   <body>
      <h1>Feedback Form</h1>

      <p>Please fill out this form to help
         us improve our site.</p>

      <!-- this tag starts the form, gives the -->
      <!-- method of sending information and the -->
      <!-- location of form script -->
      <form method = "post" action = "">
         <p>
```

Fig. 4.12 | Form with hidden fields and a text box. (Part 1 of 2.)

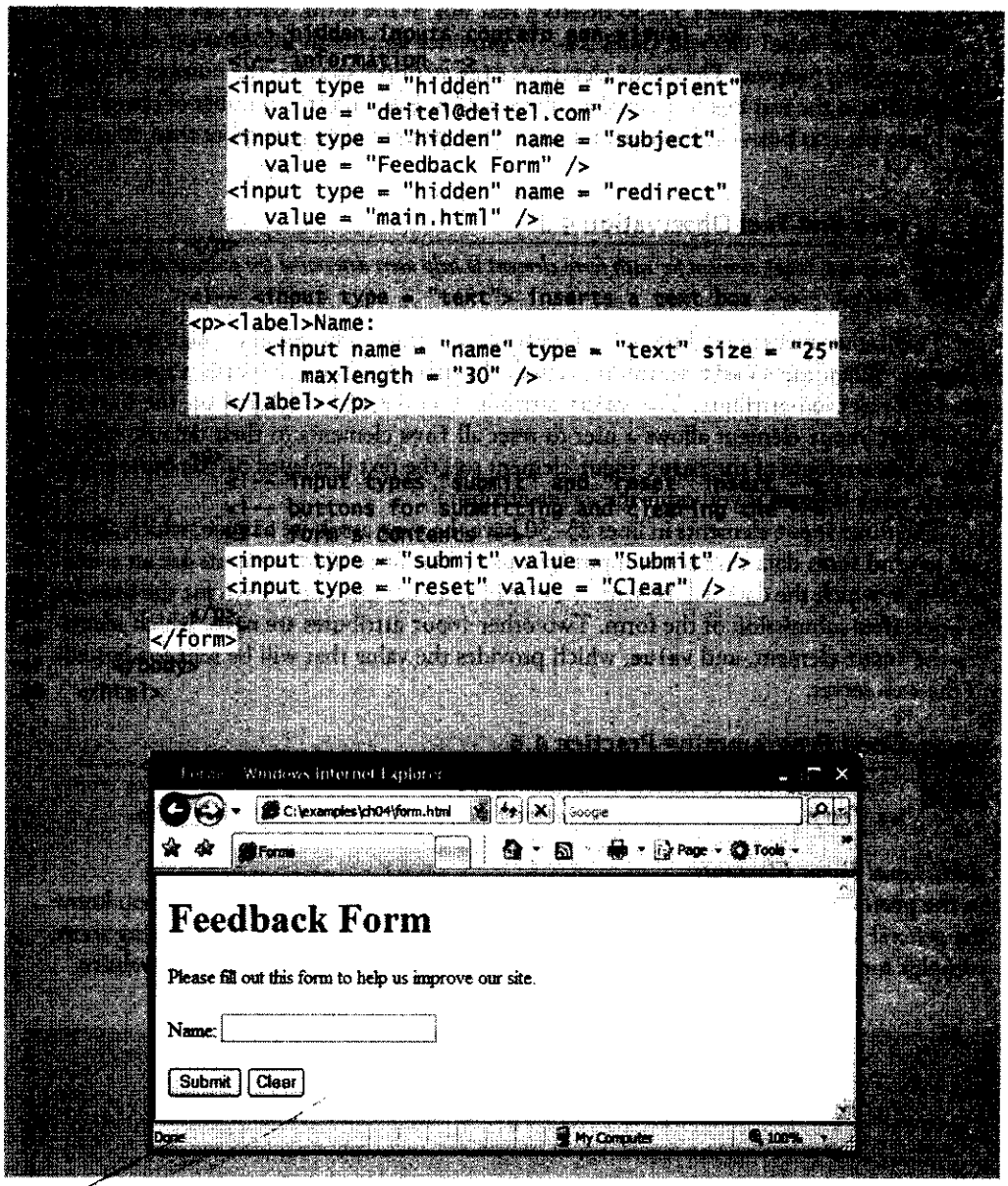


Fig. 4.12 | Form with hidden fields and a text box. (Part 2 of 2.)

The **action** attribute in the `<form>` tag in line 21 specifies the URL of a script on the web server that will be invoked to process the form's data. Since we haven't introduced server-side programming yet, we leave this attribute empty for now.

Lines 25–44 define six **input** elements that specify data to provide to the script that processes the form (also called the **form handler**). There are several types of input elements. An input's type is determined by its **type attribute**. This form uses a `text` input, a `submit` input, a `reset` input and three `hidden` inputs.

The **text** input in lines 35–36 inserts a **text box** in the form. Users can type data in text boxes. The label element (lines 34–37) provides users with information about the input element's purpose. The input element's **size** attribute specifies the number of characters visible in the text box. Optional attribute **maxlength** limits the number of characters input into the text box—in this case, the user is not permitted to type more than 30 characters.

Look-and-Feel Observation 4.3

Include a label *element for each form element to help users determine the purpose of each form element.*

Two input elements in lines 43–44 create two buttons. The **submit** input element is a button. When the submit button is pressed, the user is sent to the location specified in the form's action attribute. The **value** attribute sets the text displayed on the button. The **reset** input element allows a user to reset all form elements to their default values. The value attribute of the reset input element sets the text displayed on the button (the default value is **Reset** if you omit the value attribute).

The three input elements in lines 25–30 have the type attribute hidden, which allows you to send form data that is not input by a user. The three hidden inputs are an e-mail address to which the data will be sent, the e-mail's subject line and a URL for the browser to open after submission of the form. Two other input attributes are **name**, which identifies the input element, and **value**, which provides the value that will be sent (or posted) to the web server.

Good Programming Practice 4.6

Place hidden input *elements at the beginning of a form, immediately after the opening* <form> *tag. This placement allows document authors to locate hidden* input *elements quickly.*

Additional Form Elements

In the previous example, you saw basic elements of XHTML forms. Now that you know the general structure of a form, we introduce elements and attributes for creating more complex forms. Figure 4.13 contains a form that solicits user feedback about a website.

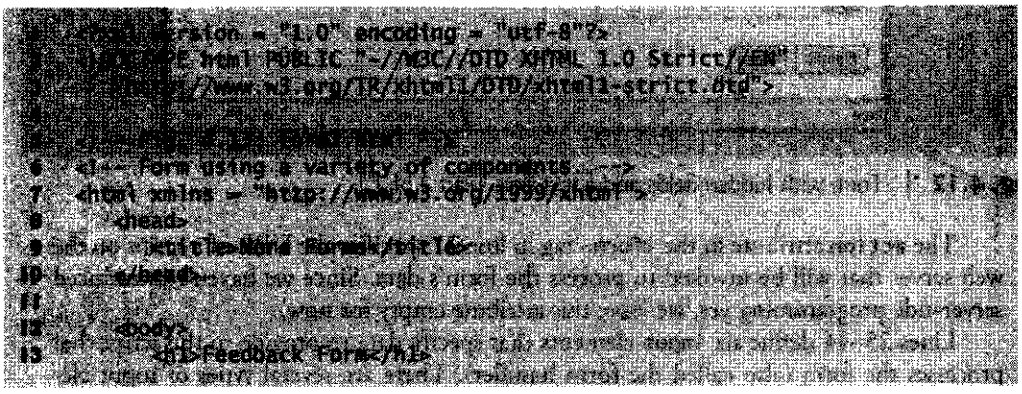
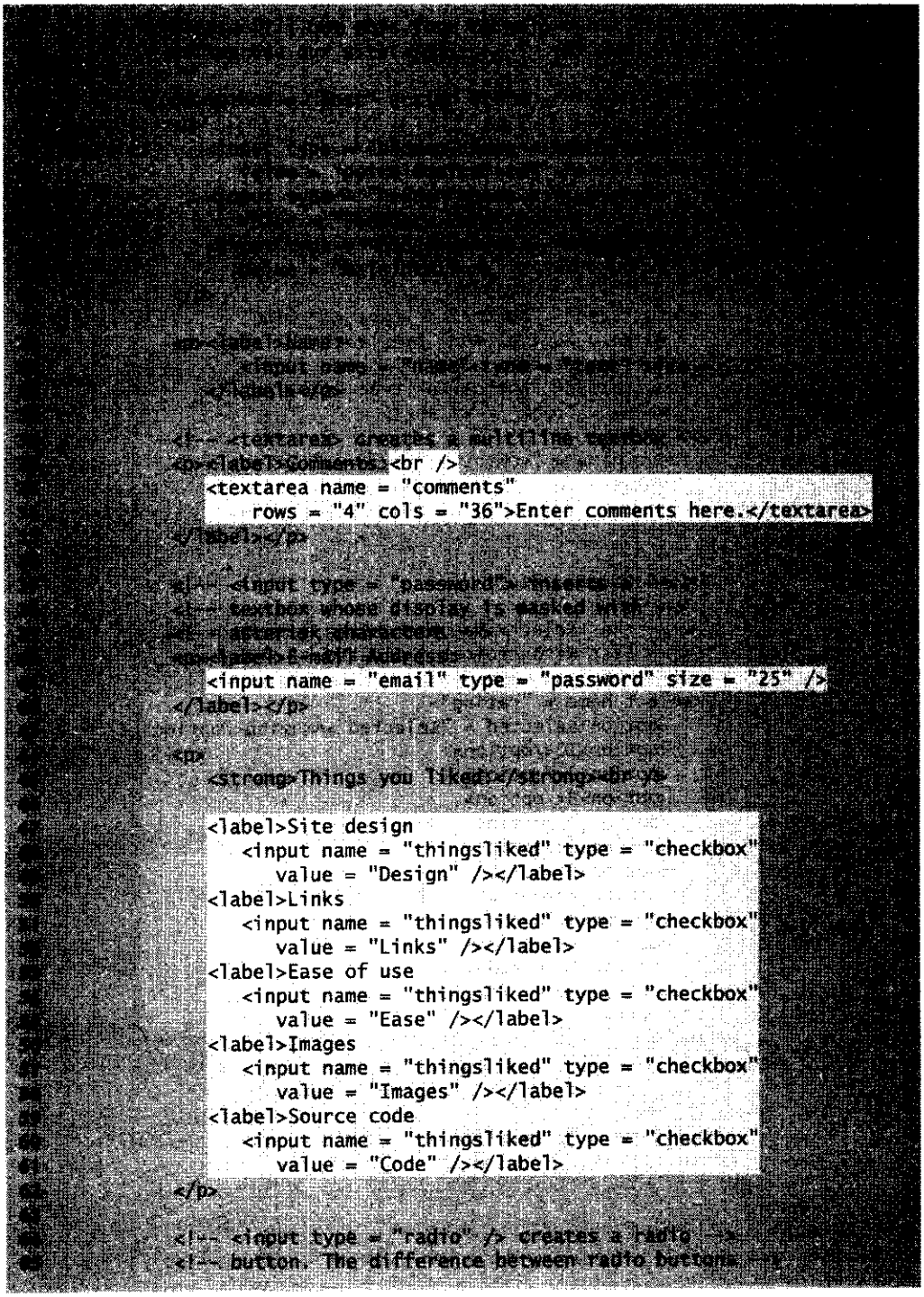
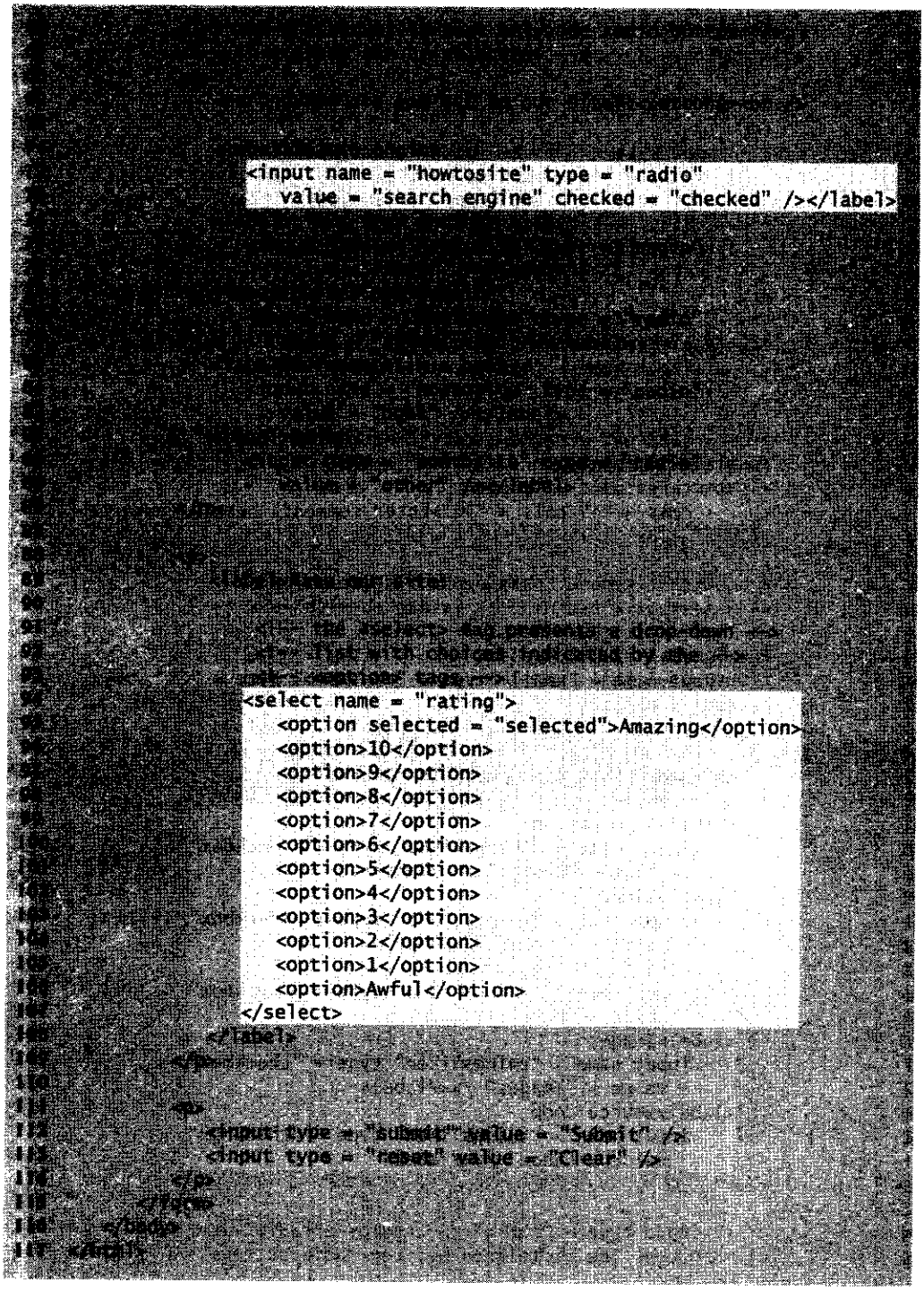


Fig. 4.13 | Form using a variety of components. (Part 1 of 4.)



```
      <textarea name = "comments"
         rows = "4" cols = "36">Enter comments here.</textarea>
      <input name = "email" type = "password" size = "25" />
      <strong>Things you liked:</strong><br />
      <label>Site design
         <input name = "thingsliked" type = "checkbox"
            value = "Design" /></label>
      <label>Links
         <input name = "thingsliked" type = "checkbox"
            value = "Links" /></label>
      <label>Ease of use
         <input name = "thingsliked" type = "checkbox"
            value = "Ease" /></label>
      <label>Images
         <input name = "thingsliked" type = "checkbox"
            value = "Images" /></label>
      <label>Source code
         <input name = "thingsliked" type = "checkbox"
            value = "Code" /></label>
```

Fig. 4.13 | Form using a variety of components. (Part 2 of 4.)



```
<input name = "howtosite" type = "radio"
   value = "search engine" checked = "checked" /></label>

<select name = "rating">
   <option selected = "selected">Amazing</option>
   <option>10</option>
   <option>9</option>
   <option>8</option>
   <option>7</option>
   <option>6</option>
   <option>5</option>
   <option>4</option>
   <option>3</option>
   <option>2</option>
   <option>1</option>
   <option>Awful</option>
</select>
```

Fig. 4.13 | Form using a variety of components. (Part 3 of 4.)

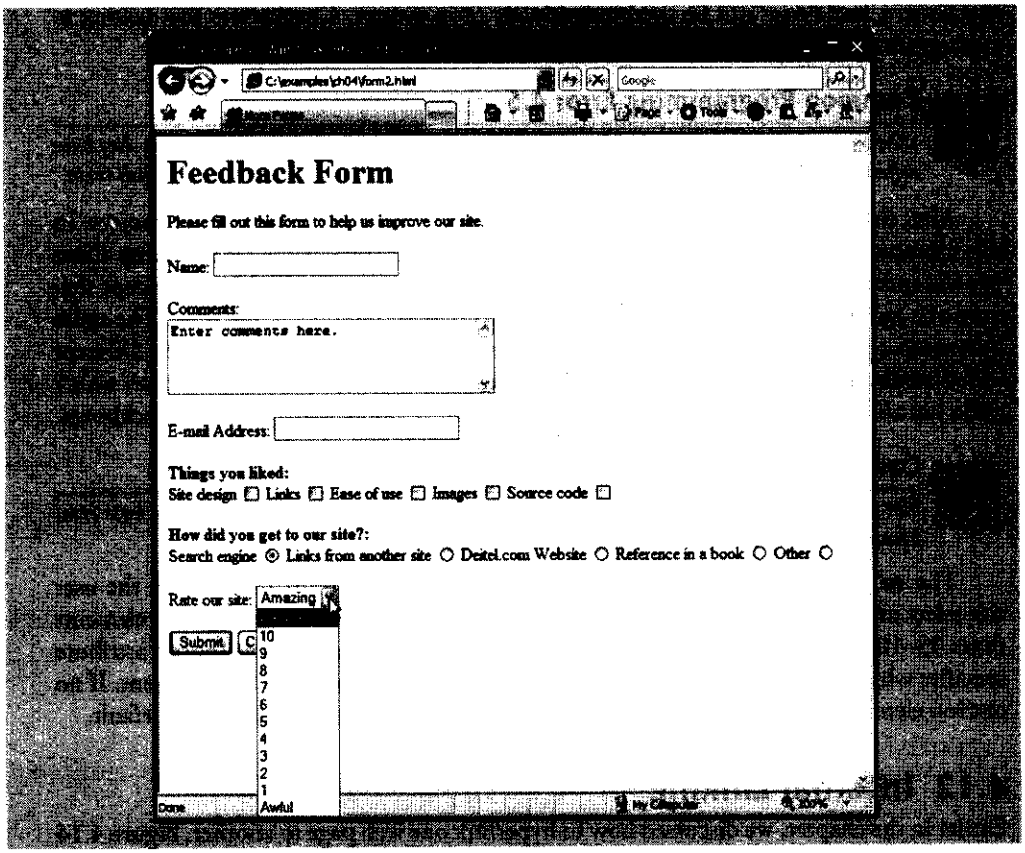


Fig. 4.13 | Form using a variety of components. (Part 4 of 4.)

In line 32, we introduce the **br** element, which most browsers render as a =**line break**. Any markup or text following a `br` element is rendered on the next line. Like the `img` element, `br` is an example of an empty element terminated with a forward slash. We add a space before the forward slash to enhance readability.

The **textarea** element (lines 33–34) inserts a multiline text box, called a **text area**, into the form. The number of rows is specified with the **rows** attribute, and the number of columns (i.e., characters per line) is specified with the **cols** attribute. In this example, the `textarea` is four rows high and 36 characters wide. To display default text in the text area, place the text between the `<textarea>` and `</textarea>` tags. Default text can be specified in other `input` types, such as text boxes, by using the `value` attribute.

The **password** input in line 41 inserts a password box with the specified `size` (maximum number of characters allowed). A password box allows users to enter sensitive information, such as credit card numbers and passwords, by "masking" the information input with asterisks (*). The actual value input is sent to the web server, not the characters that mask the input.

Lines 47–61 introduce the **checkbox** `form` element. Checkboxes enable users to select from a set of options. When a user selects a checkbox, a check mark appears in the checkbox. Otherwise, the checkbox remains empty. Each **"checkbox"** `input` creates a new

checkbox. Checkboxes can be used individually or in groups. Checkboxes that belong to a group are assigned the same name (in this case, "thingsliked").

Common Programming Error 4.5

When your form has several checkboxes with the same name, you must make sure that they have different values, or the scripts running on the web server will not be able to distinguish them.

After the checkboxes, we present two more ways to allow the user to make choices. In this example, we introduce two new input types. The first type is the **radio button** (lines 71–85) specified with type "radio". Radio buttons are similar to checkboxes, except that only one radio button in a group of radio buttons may be selected at any time. The radio buttons in a group all have the same name attributes and are distinguished by their different value attributes. The attribute–value pair checked = "checked" (line 73) indicates which radio button, if any, is selected initially. The checked attribute also applies to checkboxes.

Common Programming Error 4.6

Not setting the name attributes of the radio buttons in a form to the same name is a logic error because it lets the user select all of them at the same time.

The **select** element (lines 94–107) provides a drop-down list from which the user can select an item. The name attribute identifies the drop-down list. The **option** elements (lines 95–106) add items to the drop-down list. The option element's **selected** attribute specifies which item initially is displayed as the selected item in the select element. If no option element is marked as selected, the browser selects the first option by default.

4.12 Internal Linking

Earlier in the chapter, we discussed how to hyperlink one web page to another. Figure 4.14 introduces **internal linking**—a mechanism that enables the user to jump between locations in the same document. Internal linking is useful for long documents that contain many sections. Clicking an internal link enables users to find a section without scrolling through the entire document.

Line 14 contains a tag with the id attribute (set to "features") for an internal hyperlink. To link to a tag with this attribute inside the same web page, the href attribute of an anchor element includes the id attribute value preceded by a pound sign (as in #features). Line 56 contains a hyperlink with the id features as its target. Selecting this hyperlink in a web browser scrolls the browser window to the h1 tag in line 14. Note that you may have to resize your browser to a small window and scroll down before clicking the link to see the browser scroll to the h1 element.

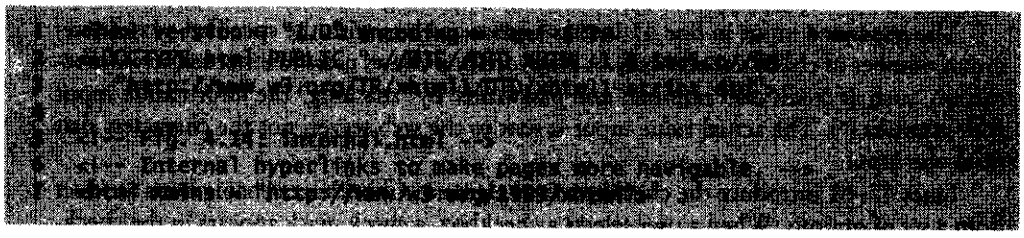
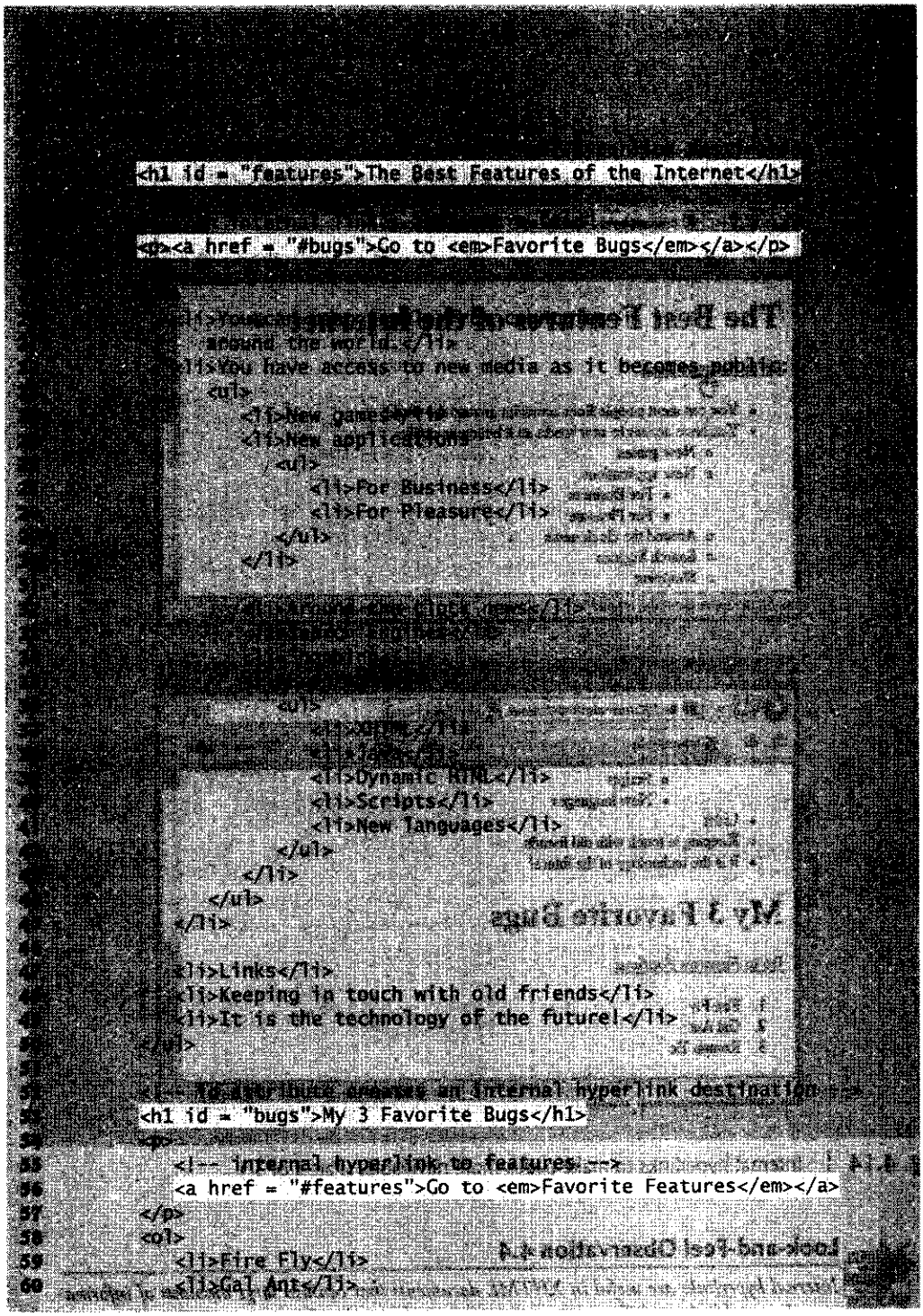


Fig. 4.14 | Internal hyperlinks to make pages more navigable. (Part 1 of 3.)



```
<h1 id = "features">The Best Features of the Internet</h1>

<p><a href = "#bugs">Go to <em>Favorite Bugs</em></a></p>

      around the world.</li>
      <li>You have access to new media as it becomes public:
         <ul>
            <li>New games</li>
            <li>New applications
               <ul>
                  <li>For Business</li>
                  <li>For Pleasure</li>
               </ul>
            </li>
            <li>Around the clock news</li>
            <li>Search engines</li>
            <li>Online</li>
            <li>Live</li>
            <li>Dynamic HTML</li>
            <li>Scripts</li>
            <li>New Languages</li>
         </ul>
      </li>
   </ul>
</li>

<li>Links</li>
<li>Keeping in touch with old friends</li>
<li>It is the technology of the future!</li>
</ul>

<!-- The following creates an internal hyperlink destination -->
<h1 id = "bugs">My 3 Favorite Bugs</h1>
<p>
   <!-- Internal hyperlink to features -->
   <a href = "#features">Go to <em>Favorite Features</em></a>
</p>
<ol>
   <li>Fire Fly</li>
   <li>Gal Ant</li>
```

Fig. 4.14 | Internal hyperlinks to make pages more navigable. (Part 2 of 3.)

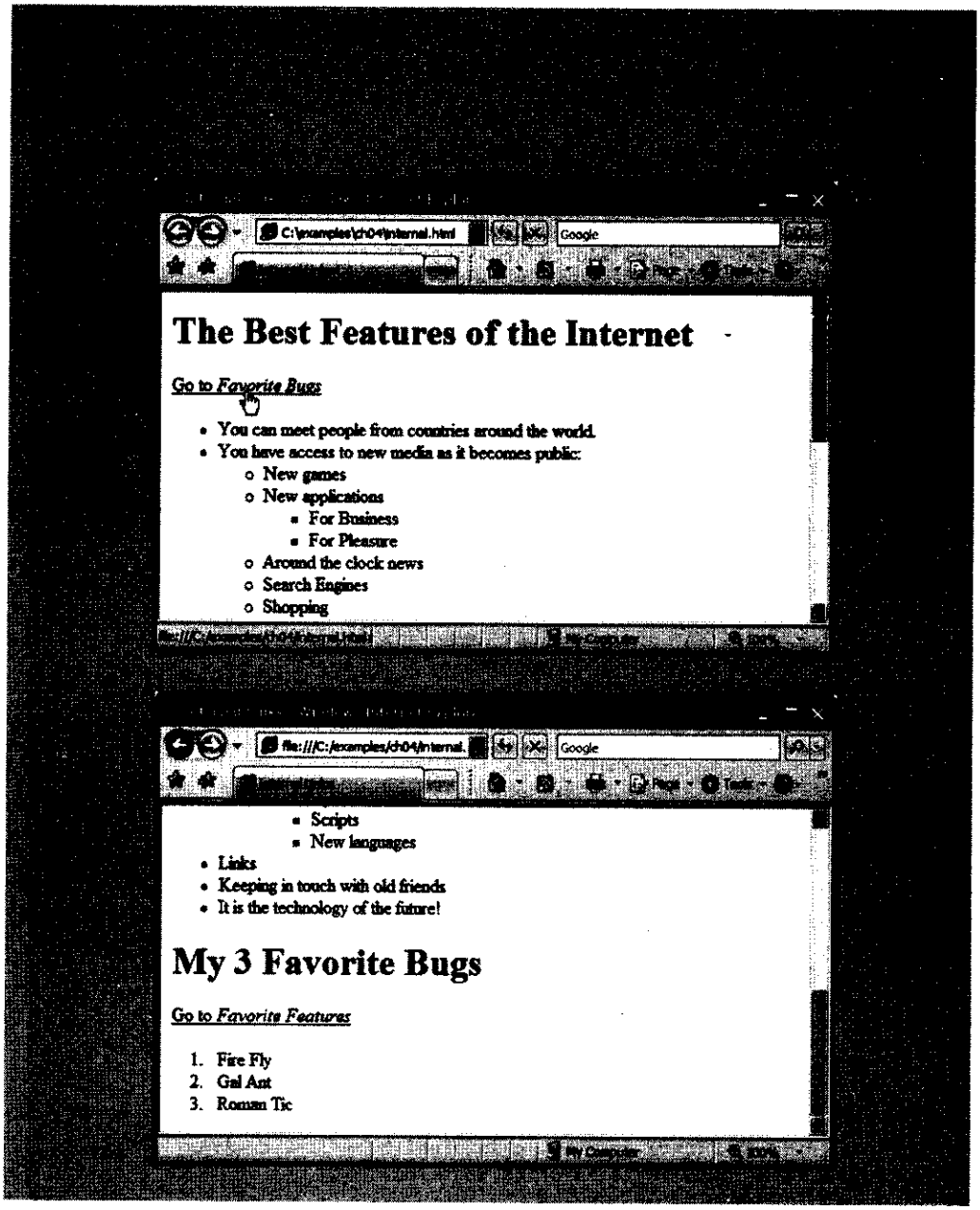


Fig. 4.14 | Internal hyperlinks to make pages more navigable. (Part 3 of 3.)

Look-and-Feel Observation 4.4

Internal hyperlinks are useful in XHTML documents that contain large amounts of information. Internal links to different parts of the page make it easier for users to navigate the page—they do not have to scroll to find the section they want.

Although not demonstrated in this example, a hyperlink can specify an internal link in another document by specifying the document name followed by a pound sign and the `id` value, as in:

```
href = "filename.html#id"
```

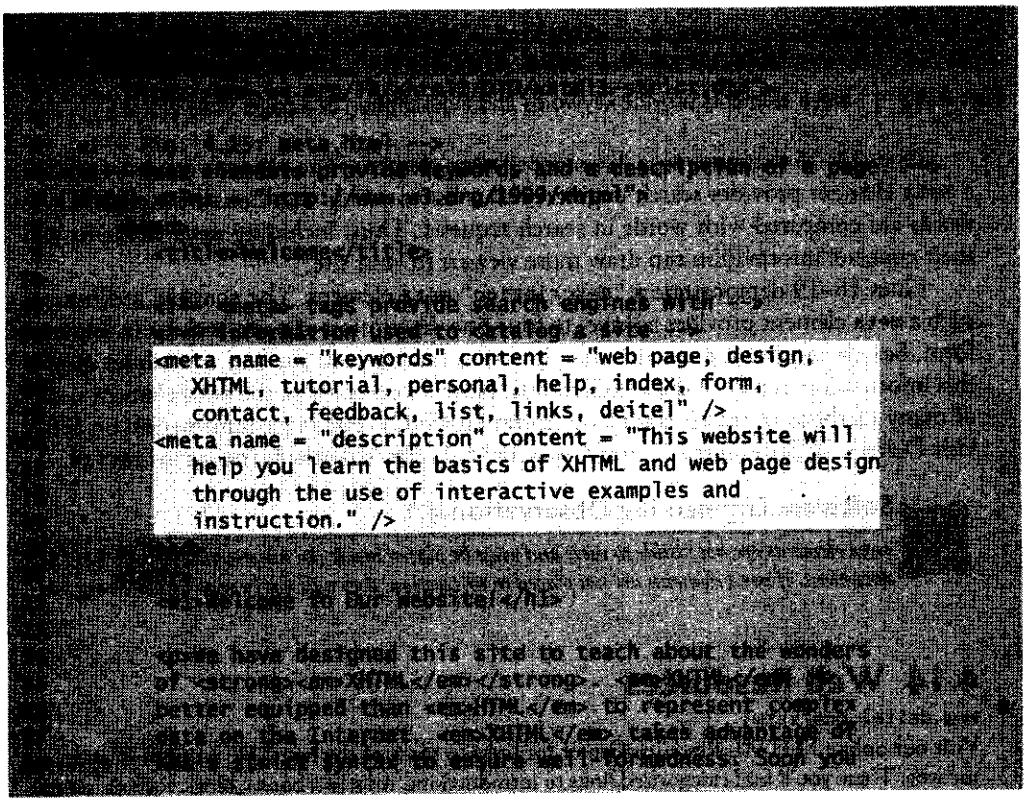
For example, to link to a tag with the `id` attribute called `booklist` in `books.html`, `href` is assigned `"books.html#booklist"`. You can also send the browser to an internal link on another website by appending the pound sign and id value of an element to any URL, as in:

```
href = "URL/filename.html#id"
```

4.13 meta Elements

Search engines help people find websites. They usually catalog sites by following links from page to page (often known as spidering or crawling) and saving identification and classification information for each page. One way that search engines catalog pages is by reading the content in each page's **meta** elements, which specify information about a document.

Two important attributes of the `meta` element are **name**, which identifies the type of `meta` element, and **content**, which provides the information search engines use to catalog pages. Figure 4.15 introduces the `meta` element.



```
<meta name = "keywords" content = "web page, design,
   XHTML, tutorial, personal, help, index, form,
   contact, feedback, list, links, deitel" />
<meta name = "description" content = "This website will
   help you learn the basics of XHTML and web page design
   through the use of interactive examples and
   instruction." />
```

Fig. 4.15 | `meta` elements provide keywords and a description of a page. (Part 1 of 2.)

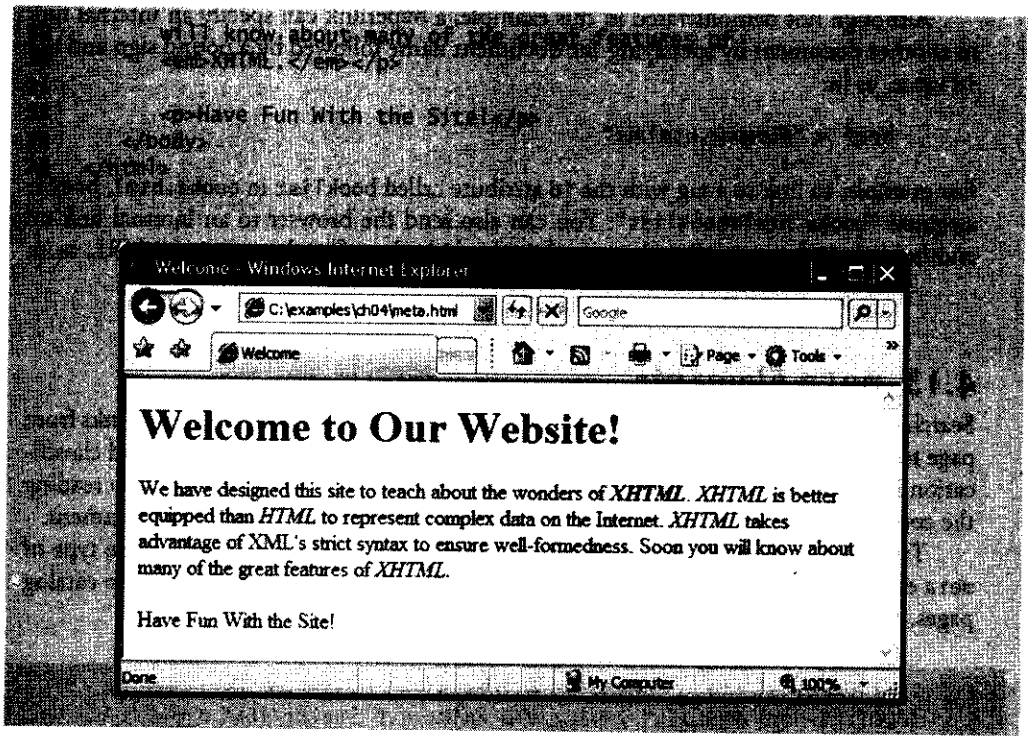


Fig. 4.15 | meta elements provide keywords and a description of a page. (Part 2 of 2.)

Lines 13–15 demonstrate a "keywords" meta element. The content attribute of such a meta element provides search engines with a list of words that describe a page. These words are compared with words in search requests. Thus, including meta elements and their content information can draw more viewers to your site.

Lines 16–19 demonstrate a "description" meta element. The content attribute of such a meta element provides a three- to four-line description of a site, written in sentence form. Search engines also use this description to catalog your site and sometimes display this information as part of the search results. Note that this use of the meta element is one of many methods of search engine optimization (SEO). For more information on SEO, visit Deitel's SEO Resource Center at www.deitel.com/searchengineoptimization.

Software Engineering Observation 4.1

meta elements are not visible to users and must be placed inside the head *section of your XHTML document. If* meta *elements are not placed in this section, they will not be read by search engines.*

4.14 Web Resources

www.deitel.com/xhtml

Visit our online XHTML Resource Center for links to some of the best XHTML information on the web. There you'll find categorized links to introductions, tutorials, books, blogs, forums, sample chapters, and more. Also check out links about XHTML 2, the upcoming version of the XHTML standard.

Summary

Section 4.1 Introduction

- XHTML (Extensible HyperText Markup Language) is a markup language for creating web pages.
- XHTML is based on HTML (HyperText Markup Language)—a legacy technology of the World Wide Web Consortium (W3C).
- XHTML 1.0 allows only a document's content and structure to appear in a valid XHTML document, and not its formatting.
- Formatting is specified with Cascading Style Sheets

Section 4.2 Editing XHTML

- A machine that runs a specialized piece of software called a web server stores XHTML documents.

Section 4.3 First XHTML Example

- In XHTML, text is marked up with elements delimited by tags that are names contained in pairs of angle brackets. Some elements may contain attributes that provide additional information about the element.
- Every XHTML document contains a start `<html>` tag and an end `</html>` tag.
- Comments in XHTML always begin with `<!--` and end with `-->`. The browser ignores all text inside a comment.
- Every XHTML document contains a `head` element, which generally contains information, such as a title, and a `body` element, which contains the page content. Information in the `head` element generally is not rendered in the display window but may be made available to the user through other means.
- The `title` element names a web page. The title usually appears in the colored bar (called the title bar) at the top of the browser window and also appears as the text identifying a page when users add your page to their list of **Favorites** or **Bookmarks**.
- The `body` of an XHTML document is the area in which the document's content is placed. The content may include text and tags.
- All text placed between the `<p>` and `</p>` tags forms one paragraph.

Section 4.4 W3C XHTML Validation Service

- XHTML documents that are syntactically correct are guaranteed to render properly. XHTML documents that contain syntax errors may not display properly.
- Validation services (e.g., `validator.w3.org`) ensure that an XHTML document is syntactically correct.

Section 4.5 Headings

- XHTML provides six headings (`h1` through `h6`) for specifying the relative importance of information. Heading element `h1` is considered the most significant heading and is rendered in a larger font than the other five headings. Each successive heading element (i.e., `h2`, `h3`, etc.) is rendered in a progressively smaller font.

Section 4.6 Linking

- Web browsers typically underline text hyperlinks and color them blue by default.
- The `strong` element typically causes the browser to render text in a bold font.
- Users can insert links with the `a` (anchor) element. The most important attribute for the `a` element is `href`, which specifies the resource (e.g., page, file, e-mail address) being linked.
- Anchors can link to an e-mail address using a `mailto:` URL. When someone clicks this type of anchored link, most browsers launch the default e-mail program (e.g., Outlook Express) to initiate an e-mail message addressed to the linked address.

Section 4.7 Images

- The `img` element's `src` attribute specifies an image's location. In a valid XHTML document every `img` element must have an `alt` attribute, which contains text that is displayed if the client cannot render the image.
- The `alt` attribute makes web pages more accessible to users with disabilities, especially those with vision impairments.
- Some XHTML elements are empty elements that contain only attributes and do not mark up text. Empty elements (e.g., `img`) must be terminated, either by using the forward slash character (/) or by explicitly writing an end tag.

Section 4.8 Special Characters and Horizontal Rules

- XHTML provides special characters or entity references (in the form *&code;*) for representing characters that cannot be rendered otherwise.
- Special character codes can be either word abbreviations or numbers, decimal or hexadecimal.
- Most browsers render a horizontal rule, indicated by the `<hr />` tag, as a horizontal line. The `hr` element also inserts a line break above and below the horizontal line.

Section 4.9 Lists

- The unordered list element `ul` creates a list in which each item in the list begins with a bullet symbol (called a disc). Each entry in an unordered list is an `li` (list item) element. Most web browsers render these elements with a line break and a bullet symbol at the beginning of the line.
- The ordered list element `ol` creates a list in which each item begins with a number.
- Lists may be nested to represent hierarchical data relationships.

Section 4.10 Tables

- XHTML tables are used to mark up tabular data. The `table` element defines an XHTML table.
- Element `summary` summarizes the table's contents and is used by speech devices to make the table more accessible to users with visual impairments.
- Element `caption` describe's the table's content. The text inside the `<caption>` tag is rendered above the table in most browsers.
- A table can be split into three distinct sections: head (`thead`), body (`tbody`) and foot (`tfoot`). The head section contains such information as table titles and column headers. The table body contains the primary table data. The table foot contains such information as footnotes.
- Element `tr`, or table row, defines individual table rows. Element `th` defines a header cell. Other data in a row is defined with `td`, or table data, elements.

- You can merge data cells with the `rowspan` and `colspan` attributes. The values assigned to these attributes specify the number of rows or columns occupied by the cell. These attributes can be placed inside any data cell or table header cell.

Section 4.11 Forms

- XHTML provides forms for collecting information from users. Forms contain visual components, such as buttons, that users interact with. Forms may also contain nonvisual components, called hidden inputs, which are used to store any data that needs to be sent to the server, but is not entered by the user.
- A form begins with the `form` element. Attribute `method` specifies how the form's data is sent to the web server.
- The `action` attribute of the `form` element specifies the script to which the `form` data will be sent.
- The `"text"` input inserts a text box into the form. Text boxes allow the user to input data.
- The `input` element's `size` attribute specifies the number of characters visible in the input element. Optional attribute `maxlength` limits the number of characters input into a text box.
- The `"submit"` input submits the data entered in the form to the web server for processing. Most web browsers create a button that submits the form data when clicked. The `"reset"` input allows a user to reset all `form` elements to their default values.
- The `textarea` element inserts a multiline text box, called a text area, into a form. The number of rows in the text area is specified with the `rows` attribute, and the number of columns (i.e., characters per line) is specified with the `cols` attribute.
- The `"password"` input inserts a password box into a form. A password box allows users to enter sensitive information, such as credit card numbers and passwords, by "masking" the information input with another character. Asterisks are usually the masking character used for password boxes. The actual value input is sent to the web server, not the asterisks that mask the input.
- The checkbox input allows the user to make a selection. When the checkbox is selected, a check mark appears in the checkbox. Otherwise, the checkbox is empty. Checkboxes can be used individually and in groups. Checkboxes that are part of the same group have the same `name`.
- The `br` element causes most browsers to render a line break. Any markup or text following a `br` element is rendered on the next line.
- A radio button is similar in function and use to a checkbox, except that only one radio button in a group can be selected at any time. All radio buttons in a group have the same `name` attribute but different `value` attributes.
- The `select` input provides a drop-down list of items. The `name` attribute identifies the drop-down list. The `option` element adds items to the drop-down list.

Section 4.12 Internal Linking

- The `a` tag can be used to link to another section of the same document by specifying the element's `id` as the link's `href`.
- To link internally to an element with its id attribute set, use the syntax *#id*.

Section 4.13 `meta` Elements

- One way that search engines catalog pages is by reading the `meta` element's contents. Two important attributes of the `meta` element are `name`, which identifies the type of `meta` element, and `content`, which provides information a search engine uses to catalog a page.
- The `content` attribute of a `keywords` `meta` element provides search engines with a list of words that describe a page. These words are compared with words in search requests.

- The `content` attribute of a `description` `meta` element provides a three- to four-line description of a site, written in sentence form. Search engines also use this description to catalog your site and sometimes display this information as part of the search results.

Terminology

`<!--...-->` (XHTML comment)
`a` element (`<a>...</a>`)
accessible web pages
`action` attribute (`form`)
`alt` attribute (`img`)
`&amp;` (`&` special character)
anchor
angle brackets (`< >`)
attribute
`body` element
`border` attribute (`table`)
`br` element (line break)
browser request
`caption` element
Cascading Style Sheets (CSS)
character entity reference
checkbox
`checked` attribute (`input`)
`cols` attribute (`textarea`)
`colspan` attribute (`th`, `td`)
comment in XHTML
`&copy;` (© special character)
data cells
database
debugging
`del` element
element
`em` element
e-mail anchor
empty element
end tag
Extensible HyperText Markup Language (XHTML)
form
`form` element
form handler
`get` request type
`head` element
header cell
heading
`h1` through `h6` (heading elements)
`height` attribute (`img`)
hexadecimal code
`hidden` `input` element
`hr` element (horizontal rule)
`href` attribute (`a`)
`.htm` (XHTML filename extension)
`<html>` tag
`.html` (XHTML filename extension)
hyperlink
`img` element
`input` element
internal linking
level of nesting
`li` element (list item)
line break
link
linked document
list item
`&lt;` (`<` special character)
`mailto:` URL
markup language
`maxlength` attribute (`input`)
`meta` element
`method` attribute (`form`)
`name` attribute
nested list
nested tag
numeric character reference
`ol` element (ordered list)
`option` element
`p` element (paragraph)
password box
pixel
`post` request type
presentation of a document
protocol
`radio` input
`reset` input
resources
`rows` attribute (`textarea`)
`rowspan` attribute (`th`, `tr`)
script
`select` element
`selected` attribute (`option`)
`size` attribute (`input`)
source code
special character

speech synthesizer
`src` attribute (`img`)
start tag
`strong` element
`sub` element
`submit` input
subscript
superscript
`table` element
tag
`tbody` element
`td` element
text editor
text-based browser
`textarea`
`textarea` element
`tfoot` element (table foot)
`thead` element (table head)
`title` element
`tr` element (table row)
`type` attribute (`input`)
`ul` element (unordered list)
valid document
validation service
`value` attribute (`input`)
value of an attribute
web page
web server
`width` attribute (`img`)
World Wide Web (WWW)
World Wide Web Consortium (W3C)
XHTML (Extensible HyperText Markup Language)
XHTML comment
XHTML document
XHTML form
XHTML markup
XHTML tag
XML declaration
`xmlns` attribute

Self-Review Exercises

4.1 State whether each of the following is *true* or *false*. If *false*, explain why.

a) An ordered list cannot be nested inside an unordered list.
b) XHTML is an acronym for XML HTML.
c) Element `br` represents a line break.
d) Hyperlinks are denoted by `link` elements.
e) The width of all data cells in a table must be the same.
f) You are limited to a maximum of 100 internal links per page.

4.2 Fill in the blanks in each of the following:

a) The _____ element inserts a horizontal rule.
b) A superscript is marked up using element _____ and a subscript is marked up using element _____.
c) The least important heading element is _____ and the most important heading element is _____.
d) Element _____ marks up an unordered list.
e) Element _____ marks up a paragraph.
f) The _____ attribute in an `input` element inserts a button that, when clicked, clears the contents of the form.
g) The _____ element marks up a table row.
h) _____ are usually used as masking characters in a password box.

Exercises

4.3 Use XHTML to create a document that contains the following text:

```
Internet and World Wide Web How to Program: Fourth Edition
Welcome to the world of Internet programming. We have provided
topical coverage for many Internet-related topics.
```

Use `h1` for the title (the first line of text), `p` for text (the second and third lines of text) and `sub` for each word that begins with a capital letter (except the title). Insert a horizontal rule between the `h1`

element and the p element. Open your new document in a web browser to view the marked-up document.

4.4 Why is the following markup invalid?

```
<p>Here is some text...
<br />
<p>And some more text...</p>
```

4.5 An image named `dettel.gif` is 200 pixels wide and 150 pixels high. Write an XHTML statement using the `width` and `height` attributes of the `img` element to perform each of the following transformations:

a) Increase the size of the image by 100 percent.
b) Increase the size of the image by 50 percent.
c) Change the width-to-height ratio to 2:1, keeping the `width` attained in part (a).

4.6 Create an XHTML document containing three ordered lists: ice cream, soft serve and frozen yogurt. Each ordered list should contain a nested, unordered list of your favorite flavors. Provide a minimum of three flavors in each unordered list.

4.7 Create an XHTML document that uses an image as an e-mail link. Use attribute `alt` to provide a description of the image and link.

4.8 Create an XHTML document that contains links to your favorite websites. Your page should contain the heading "My Favorite Web Sites."

4.9 Create an XHTML document that contains an unordered list with links to all the examples presented in this chapter. [*Hint:* Place all the chapter examples in one directory.]

4.10 State which of the following statements are *true* and which are *false*. If *false*, explain why.

a) A valid XHTML document can contain uppercase letters in element names.
b) Tags need not be closed in a valid XHTML document.
c) XHTML documents can have the file extension `.htm`.
d) Valid XHTML documents can contain tags that overlap.
e) `&less;` is the character entity reference for the less-than (<) character.
f) In a valid XHTML document, `<li>` can be nested inside either `<ol>` or `<ul>` tags.

4.11 Categorize each of the following as an element or an attribute:

a) `width`
b) `td`
c) `th`
d) `name`
e) `select`
f) `type`

4.12 Write an XHTML document that produces the table shown in Fig. 4.16.

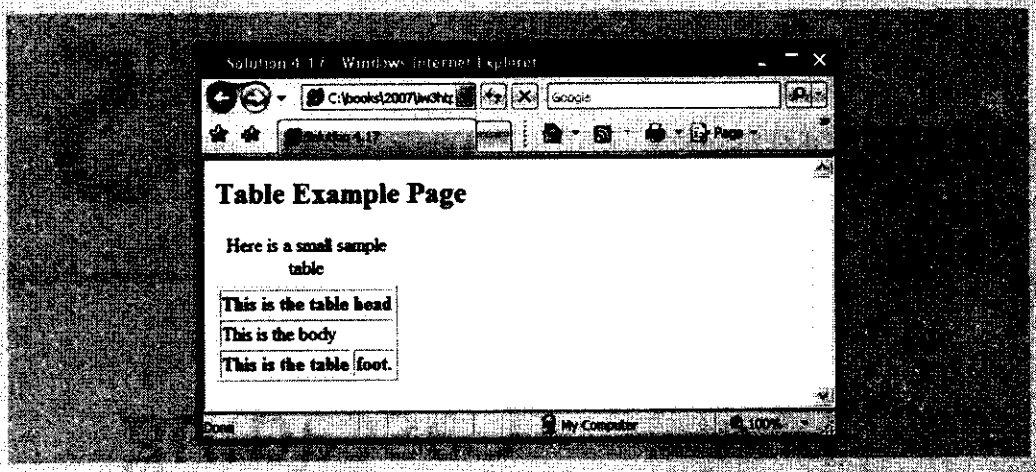
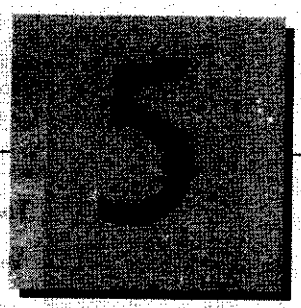


Fig. 4.16 | XHTML table for Exercise 4.12.

4.13 A local university has asked you to create an XHTML document that allows prospective students to provide feedback about their campus visit. Your XHTML document should contain a form with text boxes for a name, address and e-mail. Provide checkboxes that allow prospective students to indicate what they liked most about the campus. The checkboxes should include: students, location, campus, atmosphere, dorm rooms and sports. Also, provide radio buttons that ask the prospective students how they became interested in the university. Options should include: friends, television, Internet and other. In addition, provide a text area for additional comments, a submit button and a reset button.

4.14 Create an XHTML document titled "How to Get Good Grades." Use `<meta>` tags to include a series of keywords that describe your document.



Cascading Style Sheets™ (CSS)

Fashions fade, style is eternal.

—Yves Saint Laurent

A style does not go out of style as long as it adapts itself to its period. When there is an incompatibility between the style and a certain state of mind, it is never the style that triumphs.

—Coco Chanel

How liberating to work in the margins, outside a central perception.

—Don DeLillo

I've gradually risen from lower-class background to lower-class foreground.

—Marvin Cohen

OBJECTIVES

In this chapter you will learn:

- To control the appearance of a website by creating style sheets.
- To use a style sheet to give all the pages of a website the same look and feel.
- To use the `class` attribute to apply styles.
- To specify the precise font, size, color and other properties of displayed text.
- To specify element backgrounds and colors.
- To understand the box model and how to control margins, borders and padding.
- To use style sheets to separate presentation from content.