LEVEL 2

CHAPTER 1

Collecting Information with Forms

OBJECTIVES

In this chapter, you learn how to:

- Insert a form
- Create different types of text fields
- Insert Submit, Reset, and other buttons
- Present choices via Radio buttons and checkboxes
- Label form controls for usability and accessibility
- Create menus and lists
- Create groups of different but related controls
- Insert hidden fields

Why Would I Do This?

Forms are a very important part of the World Wide Web. They are used on many Web sites to gather information from Web-site visitors. For example, you could use a simple form to gather login information, such as username and password. Web-based e-mail, such as Hotmail and Yahoo!, is an example of a more complex use of a form. Forms can serve a wide variety of uses, such as asking questions for online surveys and polls, sending messages through ***form mail*** forms (e-mail composed and sent via a form on a Web site), querying search engines, setting preferences, posting comments on other people's Web sites, adding/deleting/editing content to database-driven Web sites, and writing content for blogs. (***Blogs*** are public online journals that are commonly used by the authors to discuss issues of interest to them. The term is a shortening of the phrase Web logs.)

Forms gather data that must be processed by a computer program (also known as a *Web application* or just *application*). These programs can be used to e-mail the message to the recipient, display the results of the search query, process your purchase order, or post your article to your blog. In the past, these types of computer applications were called *CGI applications*. (A ***CGI application***, or Common Gateway Interface program, is a program that processes information passed to it, typically through a Web form.) CGI commonly refers to applications written in Perl, C++, and other formal programming languages. (***Perl***, an acronym for Practical Extraction and Reporting Language, is a powerful, free, open-source programming language especially designed for processing text. Perl files commonly use the extension .pl or .cgi. ***C++*** is a high-level programming language commonly used to develop Windows, Macintosh, and Linux/Unix applications.) However, more recently, form data processing is performed using ASP, PHP, and other more current Web-programming languages. While you do not learn to create the processing application in this chapter, in *Chapter 8: Developing With PHP and MySQL*, you learn to create a form-processing application using the PHP language.

Forms collect information using a variety of ***form controls*** (functions or features of forms that enable the forms developer to control the type of data, amount of data, and format of data collected by the form). The most common form control is a text field in which users can type text, such as their first names. Other form controls include checkboxes from which they can make multiple choices (like the toppings they would like on a hamburger), Radio buttons that restrict them to selecting just one of the options presented (like the size of drink they want), and text areas that allow for large amounts of text (like the entire content of an e-mail message).

It is important to know how to create and use forms and form controls. Whether or not you create the application that processes the collected data, you should understand how to create a form, know the differences between the form controls, and understand the options available in forms and form controls. In some cases, you may need to consult with the Web developer to determine how the data is to be handled, and you may need to consult with a specialist in surveys or data gathering to ensure the right questions are asked and asked properly.

VISUAL SUMMARY

You use the buttons on the Forms Insert bar to create forms and insert the appropriate labels and form controls. If the Forms Insert bar is not visible, you click the Menu button at the left of the Insert bar and select Forms to switch to the Forms Insert bar.

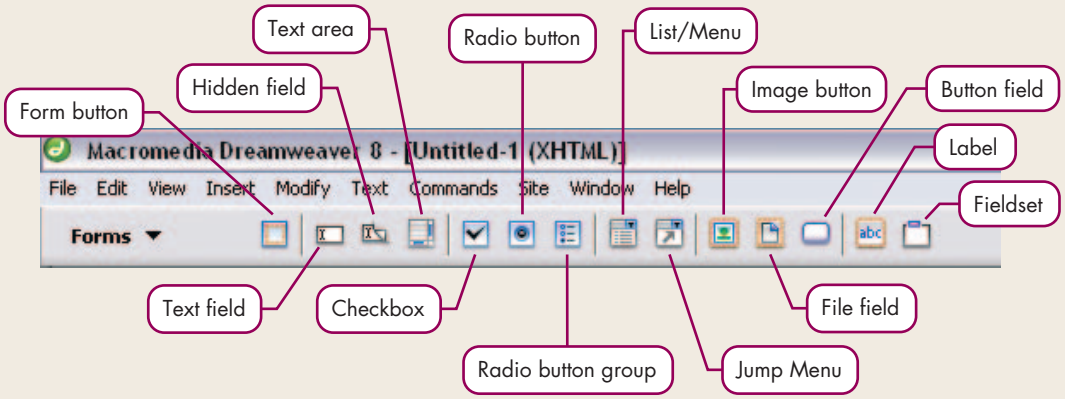

FIGURE 1.1

As with many other functions in Dreamweaver, you modify the properties of the HTML tag using the Property inspector. Forms elements are no different. After inserting a form element, you assign it a name and any other properties that are appropriate for the form or the form control.

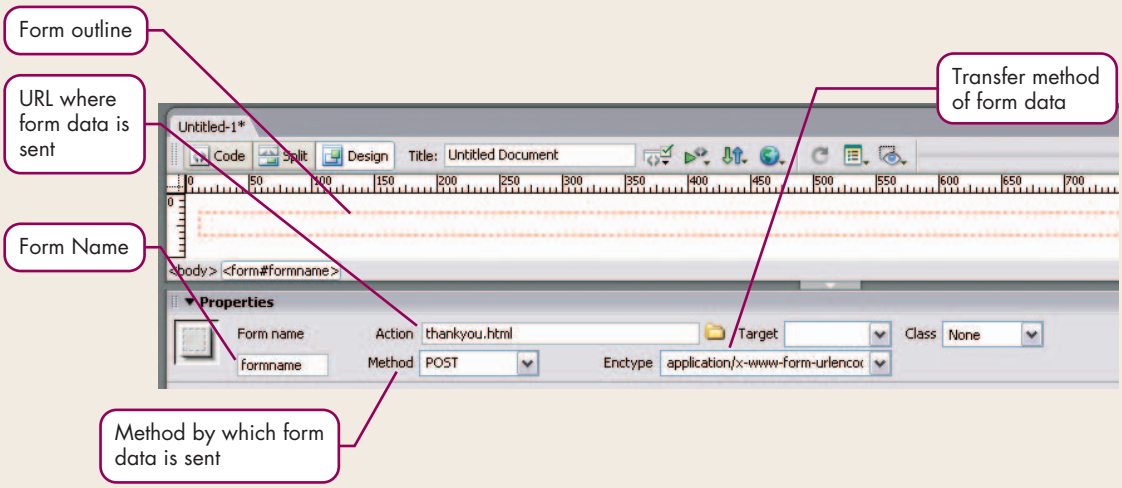

FIGURE 1.2

By default, Dreamweaver 8 provides support for Web accessibility. Form controls, such as text fields, checkboxes, and Radio buttons, have increased usability if the `<label>` tag has been applied to these controls. Each time you insert one of these controls, Dreamweaver displays the Input Tag Accessibility Attributes dialog box with which you create a label for the form control.

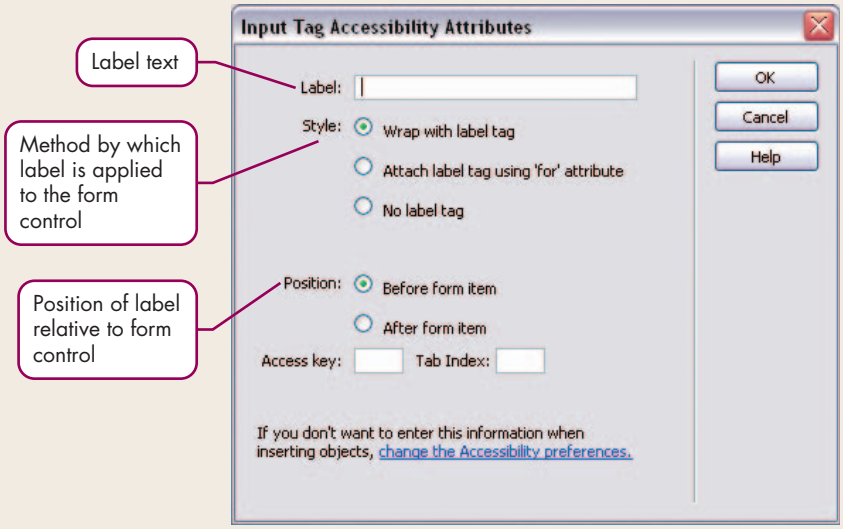

FIGURE 1.3

Text fields may be single-line or multi-line text fields. Multi-line text fields are called *text areas*. Although you can activate text fields and text areas using different buttons on the Forms Insert bar, you can also use the Property inspector for this purpose and can use it additionally to switch between the two types of text fields. Password fields are similar to single-line text fields except that the text in the field is hidden.

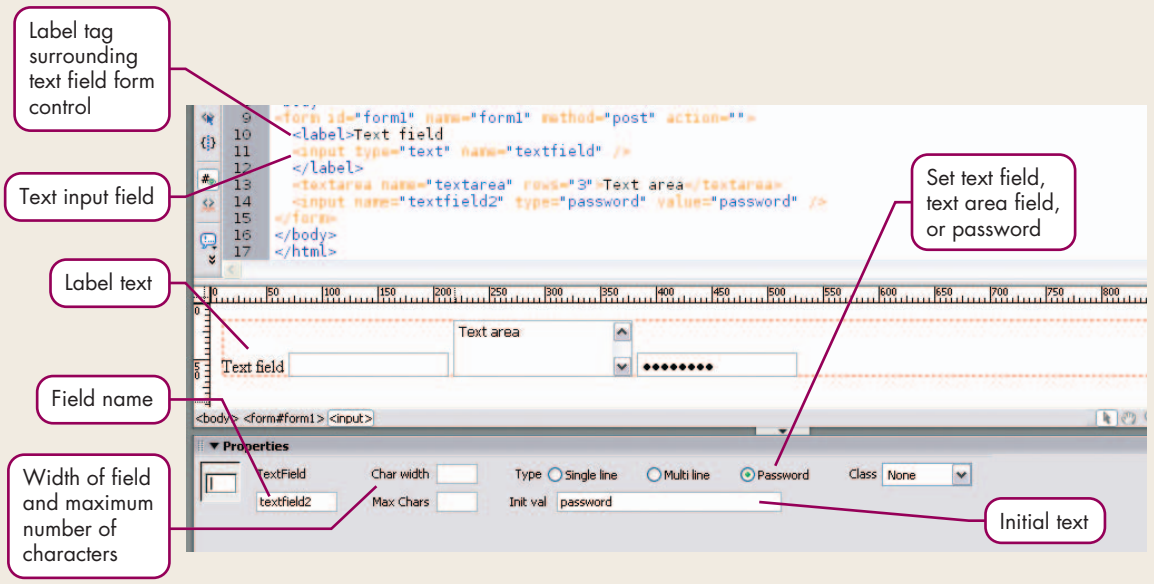

FIGURE 1.4

Checkbox and Radio buttons are similar to each other except that groups of related checkboxes allow multiple selections, such as asking the visitors to select all of their favorite sports, whereas Radio buttons allow only one answer to a question, such as selecting your gender. Checkboxes and Radio buttons commonly have their labels placed after them.

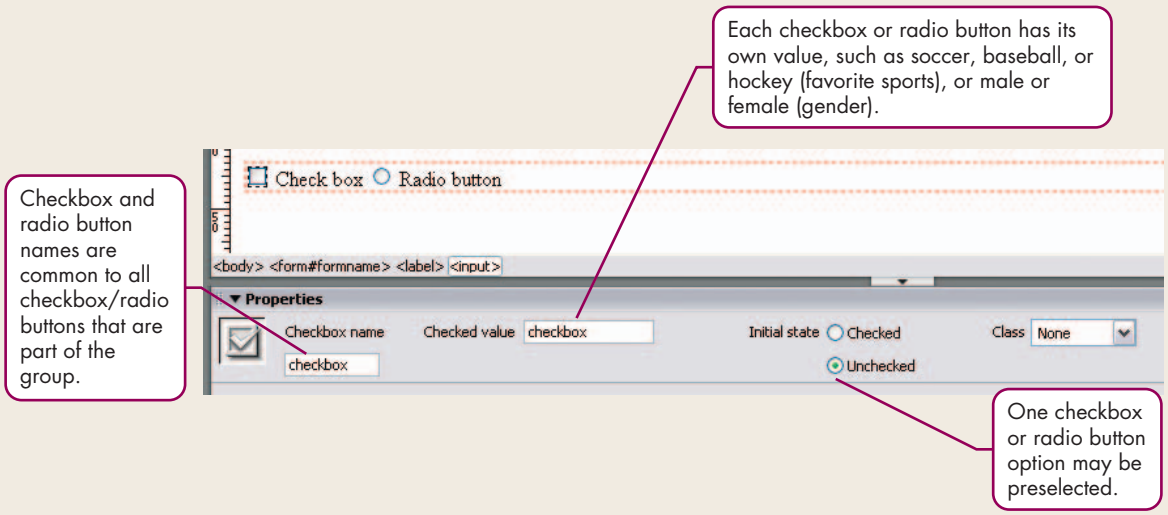

FIGURE 1.5

Menus are also form controls and are commonly used to enable a user to select a single option. Lists, which are similar to menus, allow users to make multiple selections. In addition to applying the list/menu properties in the Property inspector, you populate the list/menu with its values using a dialog box.

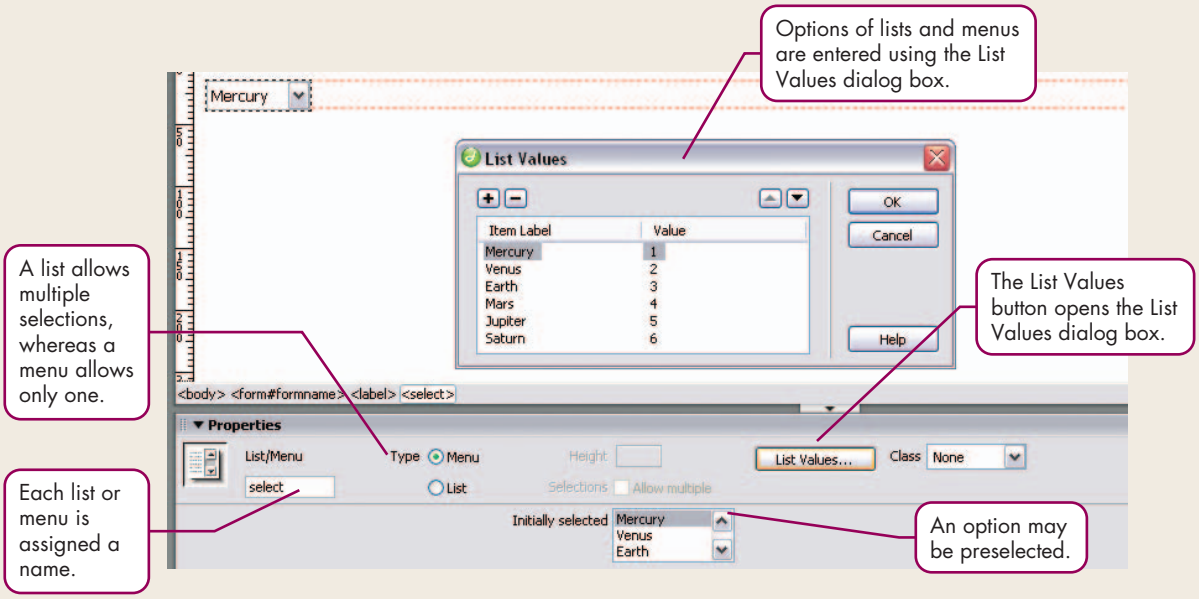

FIGURE 1.6

LESSON 1 Creating Properties of Forms

Web forms are contained between opening and closing **<form>** tags. There are three primary options required for forms: the name of the form, the method of transmitting the data, and the encoding method for the data.

The name of the form is required for the processing application. If there is only one form on a page, the name of the form is not critical. If, however, there are multiple forms on a page or if the form is part of a series of forms for either a survey or an ***e-commerce*** (electronic commerce) site, it is important that the individual forms in the series have unique IDs. (Electronic commerce can be between two businesses transmitting funds, goods, services, and/or data or between a business and a customer.)

The **action** attribute specifies the URL of the form-processing application, such as **action="contact_me.php"**. You can use either a relative or an absolute path to the processing application. Some basic form-processing software can be ***remotely hosted*** (existing on another Web server), in which case, you would need to add http://www.remote-host.com/ to the absolute path of the action URL. On some Web sites, you see the action URL with a cgi-bin folder, such as **action="/cgi-bin/formmail.pl"**. The cgi-bin folder is commonly below the root folder of the Web site for security reasons to prevent unauthorized access or use of any applications that may exist in the folder. The *mailto:* URL, such as **action="mailto:customer@domain.com"**, is a valid action URL but is rarely used and poorly supported by browsers. Its use occasionally opens a visitor's e-mail software, which can be confusing.

The **method** attribute has two options, **post** and **get**. Search engines commonly use the *get* method — the data from the form are displayed in the URL sent to the search engine. For example, when searching Google.com for php editor, the *get* URL is http://www.google.com/search?q=php+editor&sourceid=firefox&start=0&start=0&ie=utf-8&oe=utf-8. The processing application then takes the ***query string*** (the portion of the action URL after the ?), processes it, and displays the results. It is just as easy to copy this URL from the browser's address bar as it is to bookmark the URL should you wish to return to a specific search later. The *get* option limits the number of characters that may be sent: a blog or news entry sent via the *get* method may exceed the character limit (which varies with operating system, Web-server software, and the processing application). The *get* option also may pose a security risk if sensitive information is revealed, such as credit card information. Furthermore, an unscrupulous user may create problems for the processing application by modifying a *get* URL, replacing part of it with other information. On the other hand, when you develop a form-processing application, you may want to use the *get* option to watch how the information is being sent so that you can adjust your programming code accordingly.

The *post* method does not display the query string, but the data is formatted identically. The *post* method is more suitable for large amounts of data because it does not limit the amount of data sent. Furthermore, because the form data is not exposed in the URL, it is more secure and may also be ***encrypted*** (translated into a secret code) using the ***HTTPS protocol*** (also written as S-HTTP, a protocol for securely transmitting data between a browser and a server) that is used by e-commerce applications.

The **enctype** attribute specifies the type of data being transmitted. The **application/x-www-form-urlencoded** option transmits text. Using this option, spaces between words are replaced with + and certain reserved characters are replaced with hexadecimal codes for the characters. Line breaks are replaced with **%0D%0A**. The other option, **multipart/form-data**, is reserved for use when files are being ***transmitted*** (uploaded) via the form: the **method** attribute must be set to **post** when a file is being uploaded.

Create and Set the Form Properties

In this exercise, you prepare a basic form — the login form for returning customers. You learn to set up the form options using the Forms Insert bar and the Property inspector.

1. Define the Chapter_01>Forms folder as a site called `Forms`.
2. Open login.html and click in the empty cell between the header and footer cells.
3. Type `Login to Place Your Order` and format it as Heading 2.
4. Switch the Insert bar to Forms.
5. Click the first button on the left of the Forms Insert bar.

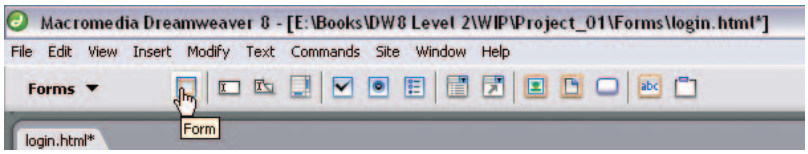

FIGURE 1.7

6. In the Property inspector, type `login` in the Form Name field, type `confirm-login.html` in the Action field, set the Method to Get, and set Enctype to `application/x-www-form-urlencoded`.

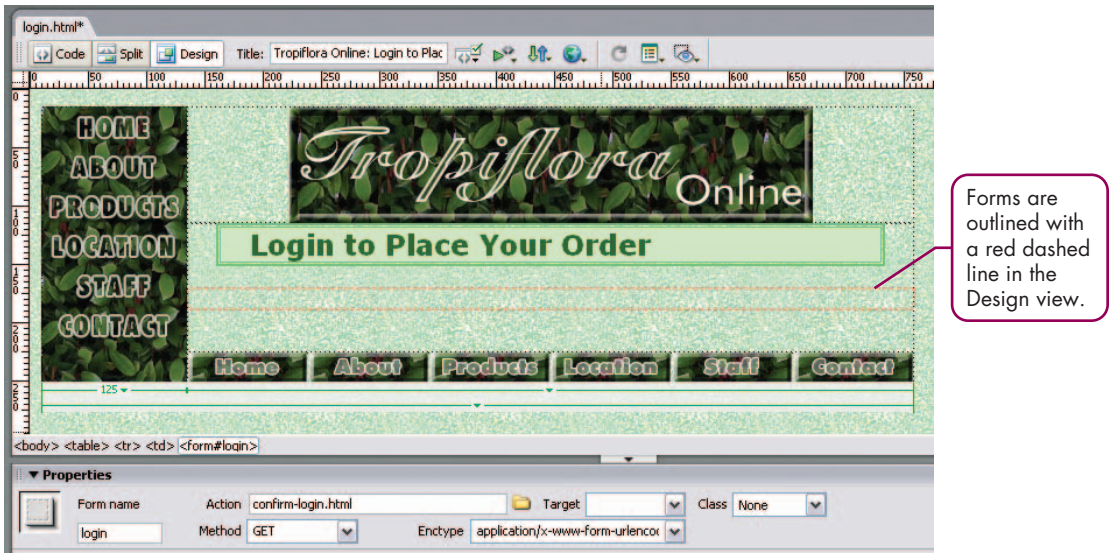

FIGURE 1.8

7. Save the changes but leave the file open for the next exercise.

To Extend Your Knowledge . . .

FREE FORM-PROCESSING APPLICATIONS

Many of the common form-processing applications are available for free. HotScripts.com is an example of an archive that offers many programs for processing form mail. If you need to perform client-side form validation, you would search in the client-side languages, such as JavaScript or VBScript (supported only by Internet Explorer). If you need to perform server-side processing of form data and your needs are common (such as form mail), search in the languages you know your server supports, such as PHP, ASP, ASP.Net, or Perl. Generally, these programs are well documented: they come with instructions on how to install and configure them for your specific needs. Why go to the trouble of trying to learn a programming language when you can just download one and use it for free?

Not all scripts on HotScripts.com and other scripts archives are free: some are feature-limited trial versions; others are commercial; some are free for personal but not commercial use; and some are free, but to remove the script-author's Web-site link or logo, you must pay for the script. You must purchase complex or unique scripts either by downloading and paying for the script or by hiring a developer to create a script for your Web site.

Bravenet.com and other sites offer free, hosted, form-processing scripts. After subscribing to Bravenet.com, you are provided with the URL of the form-processing script (for the **action** attribute) and other information required to set up the form. When a visitor uses your form, the data is sent to the form-processing script on Bravenet.com and then the data is e-mailed to you.

LESSON 2 Creating Short-Text, Long-Text, and Password Form Controls

One of the most commonly used form controls is the text field. Most form controls use the `<input />` tag, which employs the `type` attribute to specify the type of input. For text fields, for example, you set `type` to text, such as `type="text"`. The `size` attribute specifies the width of the text field in characters; this field is named *Char Width* in the Property inspector. In Dreamweaver, the default width is approximately 24 characters, but if you do not specify a width, browsers use their own default widths for the text field. The `maxlength` attribute (the Max Chars field in the Property inspector) specifies the number of characters allowed in the text field. If you do not specify a default number, the number of characters allowed is virtually unlimited, but the width of the field makes it difficult to use long text. If the size of the field is smaller than the maxlength, the field contents scroll to allow the user to enter as much text as the maxlength allows. However, if the size is larger than maxlength, space appears to the right of the contents of the field.

All form controls require a name. Although generally the **name** attribute is deprecated in XHTML 1.0 and replaced by the **id** attribute, in the case of form controls, the **name** attribute is not deprecated. The form-processing application matches the submitted data to the applicable field using the **name** attribute. Regardless of whether the *get* or *post* method is selected for the **<form>** tag, the form-control name and its value are paired in the submitted data using the format name=value, such as **firstname=Thomas&lastname=Denon**. The **id** attribute can also be applied to the form controls for either CSS or JavaScript. Dreamweaver, however, does not provide an ID option in the Property inspector for text fields, just a class option that may also be used by CSS and JavaScript.

The Property inspector also provides an Init Value field, which is equivalent to the **value** attribute in HTML. Any text you type into the Init Value field displays in the field. You may use this option to suggest to users what they may enter into the field or in what format. For example, you could set the initial value of a comments field to **Type your comments here**, or in a telephone field, you could set the initial value to **(555) 555-5555** to suggest the expected format. There is a useful JavaScript function that you can employ when using an initial value — as soon as the user types in the field, the initial value is erased, leaving only what the user types.

In accordance with Dreamweaver 8's enhanced accessibility features, each time you insert a form control, Dreamweaver prompts you for a label for the field. The purpose of a label is to describe the contents of the field. Most Web designers, whether or not they understand Web accessibility, would insert a label as text beside the form controls anyway. However, when a label is created via Dreamweaver's Input Tag Accessibility Attributes dialog box, the **<label>** tag is applied. The use of the **<label>** tag is discussed in greater detail in *Lesson 7: Labeling Form Controls*.

Create a Single-Line Text Field

In this exercise, you use two Dreamweaver options for the Text field: the Single-Line option and the Password option. The Single-Line option sets the **type** attribute of the **<input>** tag as **type="text"**, and the Password option sets the **type** attribute as **type="password."** The Multi-Line option creates a different type of form control using a different HTML tag, the **<textarea>** tag, which you use in a later exercise.

1 **In the open login.html page, click inside the form area.**

2 **From the Forms Insert bar, click the Text Field button.**

3 **Type `Customer Number:` in the Label field. Select Wrap with label tag from the Style options, select Before form item from the Position options, leave the Access key and Tab index fields blank, and click OK.**

FIGURE 1.9

4 **In the Document window, click the new text input field (white box).**

5 **In the Property inspector, select Single line from the Type options, type `cust_num` in the Text Field Name field, type `15` in the Char Width field, type `8` in the Max Chars field, type `123-4567` in the Init Value field, and press Enter/Return.**

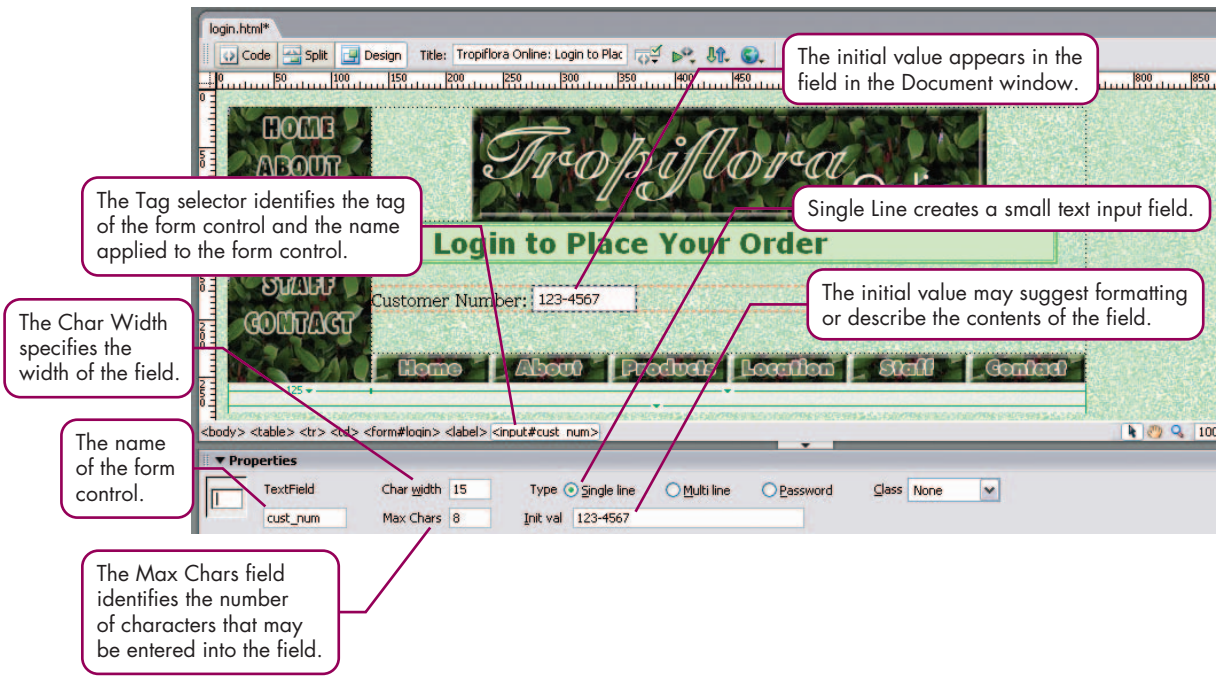

FIGURE 1.10

6 Click to the right of the text field in the Document window, and press Enter/Return.

7 Click the Text Field button in the Forms Insert bar.

8 Type `Password:` in the Label field of the Input Tag Accessibility Attributes dialog box, leave the rest of the options as they are, and click OK.

9 In the Document window, click the new input text field. In the Property inspector, set the Text Field name to `password`, the Char Width to `10`, Max Chars to `6`, and Type to Password. Type `abcdef` in the Init Val field and press Enter/Return.

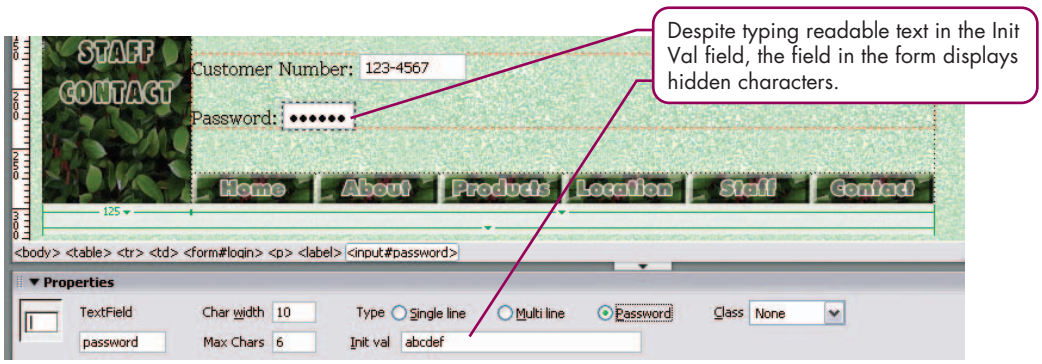

FIGURE 1.11

10 Delete the text in the Init Val field and press Enter/Return to apply the change.

11 Close login.html, saving your changes.

To Extend Your Knowledge . . .

SETTING DIMENSIONS USING CSS

You may wonder why the text, password, and other form controls use characters as the measurement for width. This measurement is a throwback to early Web browsers that used a monospaced font like Courier as the common font. With the width set to a specific number of characters, users could see exactly how much space they were allowed for that particular field. Most current browsers use proportionally spaced fonts, in which iii is narrower than MMM, although both are three characters. However, you may use CSS to specify the width of form controls, such as in pixels or ems (the width of the letter M). If you want to specify different widths for different form controls, you must apply unique IDs to each control so that you may specify the width for just that control. If you want several controls to be the same width, you can apply the same CSS class to each, and then they may all use the same width.

To Extend Your Knowledge . . .

EXTRA PARAGRAPH BELOW FORMS

Most of the time, when you create a form in Dreamweaver, an extra empty paragraph appears below the form. If you wish to remove it, click in the paragraph in the Document window, click the `<p>` tag in the Tag selector, and press Delete/Backspace.

LESSON 3 Working with Multi-Line Text Areas

Unlike many of the form controls, the multi-line text area does not use the **`<input />`** tag but, instead, uses the **`<textarea>`** tag. Dreamweaver puts both single-line and multi-line form controls in the same Property inspector panel, despite the difference in tags. However, unlike the **`<input />`** tag, the **`<textarea>`** tag is a container tag requiring the closing **`</textarea>`** tag. Just like the single-line text control, **`<textarea>`** can be set with an initial value that is recorded in the Init Val (initial value) field in the Property inspector. The initial value, however, is written between the opening and closing **`<textarea>`** tags, such as, **`<textarea>I would like to hear from you.</textarea>`**, unlike any of the **`<input />`** form controls, which use the **`value`** attribute (the **`value`** attribute does not exist for the **`<textarea>`** tag).

You set the width of the text area with the Char Width field in the Property inspector; this information is applied to the **`cols`** attribute. You set the height of the text area with the Num Lines field of the Property inspector; this information is applied to the **`rows`** attribute. The units of the width measurement are characters and the units of height are lines.

Developers commonly use the **`<textarea>`** tag to create an area in which users can enter large amounts of text. Dreamweaver's Property inspector offers the Wrap option with four possible values: Default, Off, Virtual, and Physical. However, the wrap attribute of the **`<textarea>`** tag is confusing for many reasons — it was never part of any HTML standard; some HTML resources substitute Soft in place of Virtual and Hard in place of Physical; and current browsers treat Soft, Hard, Virtual, and Physical identically. Given the potential confusion and lack of validity, we recommend that you leave the setting at Default.

Create a Multi-Line Text Area

In this exercise, you create a basic form to which you add a text area.

1. **Open comments.html from the Forms site and click within the form area (red dashed line).**
2. **Click the Text Field button on the Forms Insert bar.**
3. **Type `Comments:` in the Label field and click OK.**
4. **Click the input text field in the Document window, and in the Property inspector, select Multi-line from the Type group of options.**

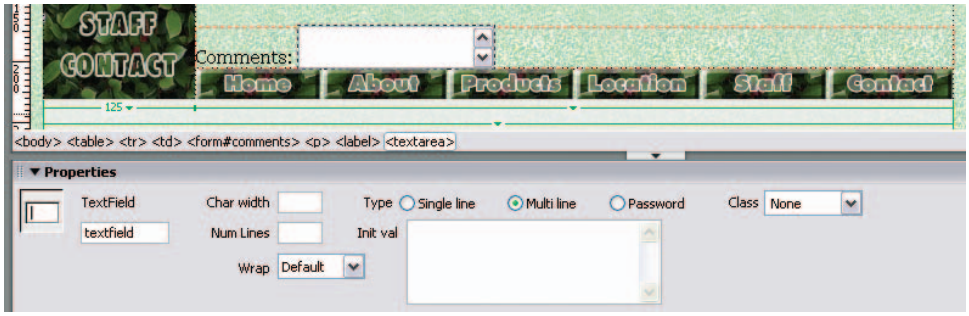

FIGURE 1.12

5 **Press Control/Command-Z (Undo) twice. Click the Textarea button on the Forms Insert bar, type `Comments:` in the Label field, and click OK.**

Notice that the result is the same: the Textarea button simply preselects the Multi-Line option.

6 **Type `comments` in the Textarea Name field, set Char Width to `35`, set Num Lines to `8`, in the Init Val field, type `Type your comment here.`, and press Tab.**

Do not press Enter/Return while the insertion point is in the Init Val field or you will create a new line in the field.

7 **To clean up some of the formatting, click between Comments: and the textarea and press Shift-Enter/Return.**

8 **In the bottom half of the Property inspector, set the horizontal alignment of the cell to center.**

FIGURE 1.13

9 **Close comments.html, saving your changes.**

To Extend Your Knowledge . . .

FORM CONTROLS OUTSIDE THE WWW

Form controls were not first brought into being when HTML was created but existed long before. Examine any computer application on your computer and you will recognize many of these controls in Web pages. Menus, Radio buttons, checkboxes, labels, single-line and multi-line text fields, and password fields all existed in computer applications before Sir Tim Berners-Lee wrote one character of HTML. If you are having trouble figuring out which controls to place in a Web form, try to think of an example in a computer application and use that as your inspiration.

LESSON 4 Using Submit, Reset, Image, and Other Buttons

The purpose of the Submit button is to send data from the completed form to the processing application. The Reset button resets the values of fields to their initial values. As with text and password form controls, you create buttons using the **`<input />`** tag with different types: for a Submit button, **`type="submit"`**, and for a Reset button, **`type="reset"`**. These types have special meaning to the browser, which submits or resets the data depending on which button the user clicks.

Two other attributes to use with these two buttons are the **`name`** and **`value`** attributes. When you select Submit or Reset from the Property inspector in Dreamweaver, the **`name`** attribute, by default, is set to that option. However, if the purpose of your form is to enable users to edit or delete items in their shopping carts, you might make the names of the buttons Edit and Delete, and then you must program the form-processing application to pick up the name of the button selected and process the data appropriately.

You can use the **`value`** attribute to replace the text on the button. In Dreamweaver, you use the Label field in the Property inspector to enter the value-attribute content. For example, rather than Submit on a contact form, you might want to use **`value="Send Me Your Comments"`**, and in place of Reset, you might want to use **`value="Clear the Form"`**. The **`value`** attribute does not change the Submit and Reset buttons' functions — the **`type`** attribute specifies these buttons' functions to the browser.

If you want to provide Edit and Delete buttons on the form so users can edit their addresses when they move or delete items from their shopping carts, in both cases you must set the **`type`** attribute to **`submit`**, because these actions must be provided by the form-processing application. The names could be Edit and Delete, and the values could be Edit This Address and Remove This Item. Therefore, if you want to provide a function that requires the form-processing application to act, you set **`type="submit"`** whatever the name or value you assign to that button.

Dreamweaver also provides a None option, which creates a generic button. You may assign it a name and value, but unlike the Submit type, clicking it does not submit the form data to the action URL, nor does the browser perform a client-side action to the data as it does with the Reset type. The purpose of the None type (in HTML, it is coded as **`type="button"`**) is to allow a Web developer to assign a JavaScript function to the button, which, when clicked, triggers that JavaScript function.

The button labeled Image Field on the Forms Insert bar inserts an image form control. This form control also uses the `<input />` tag with the `type` attribute set to `image`. An Image Field behaves like a Submit button, and, commonly, an Image Field is simply graphic text, such as Place Your Order. Like all images, you should assign alternative text. Given that most Image Field buttons replace a plain Submit button, the alternative text should be the same as the text that appears in the graphic. However, it is possible to use the image of a product as an image field that submits the customer's intention to purchase that product. A Web page with many product images could therefore enable customers to submit order requests using any of the product-image image fields.

By default, Dreamweaver 8 will prompt you to add a label to all form controls. However, screen-reading software can read the text of a button so a label is not necessary and you may cancel the Input Tag Accessibility Attributes dialog. As mentioned above, image buttons do require alternate text, so be certain to add it.

In this exercise, you add both Submit and Reset buttons to your form. You preview the form in your browser, test the functionality of the buttons, and examine the query string sent to the action URL.

Add Submit and Reset Buttons

1. **Open login.html, click to the right of the password field, and press Enter/Return.**
2. **Click the Button button (third from the right end of the Forms Insert bar).**
3. **Cancel the Input Tag Accessibility Attributes dialog box: a label is not necessary for a button.**
4. **Click to the right of the Submit button, insert another button, and cancel the Input Tag Accessibility Attributes dialog box.**

   By default, Dreamweaver creates a Submit button.
5. **In the Property inspector, change the button name to `Reset`, change the Value field to `Clear this form`, and press Enter/Return. Change the Action to Reset form.**

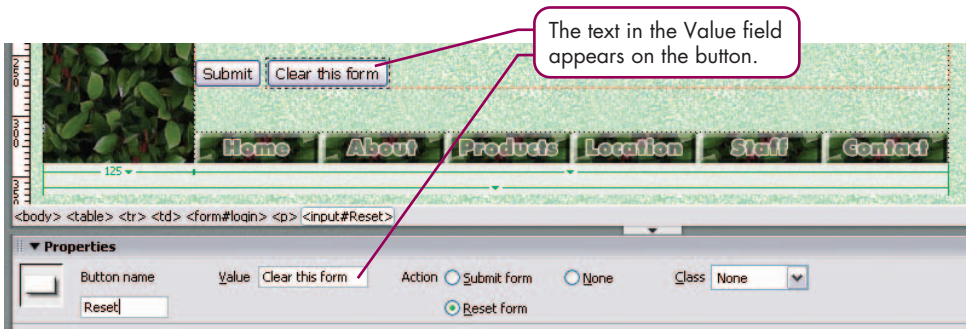

FIGURE 1.14

6 **Save the changes and preview this page in your browser.**

7 **Delete the Customer Number initial value, and type as much text as you can.**

The Max Chars property (maxlength attribute) prevents you from entering more than eight characters.

8 **Enter some text in the Password field.**

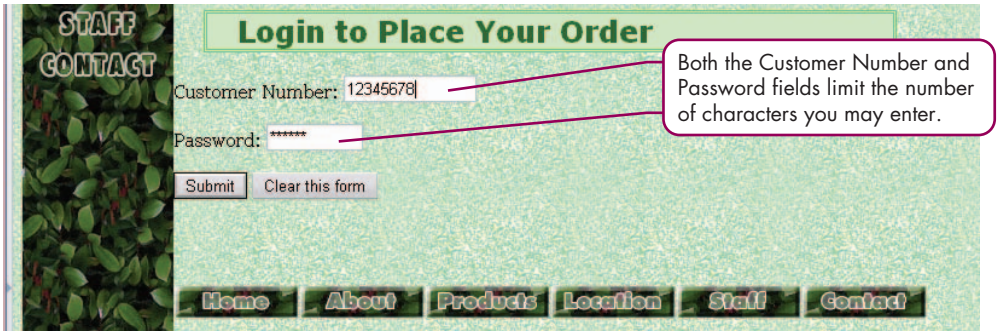

FIGURE 1.15

9 **Click the Clear This Form button.**

Both (all) fields are reset to their initial values.

10 **Again insert a fictitious customer number and password in the form and click the Submit button.**

This time, the confirm-login.html page opens in your browser because your browser has directed you to the action URL.

11 **Click in the address bar of your browser (if necessary, press the End key to view the end of the URL).**

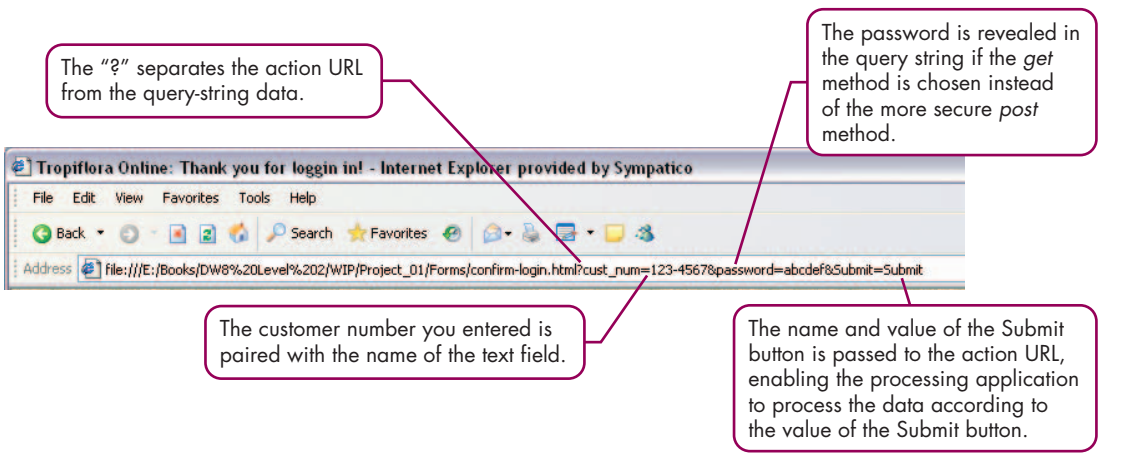

FIGURE 1.16

12 **Close your browser, return to Dreamweaver, and close login-html.**

To Extend Your Knowledge . . .

THE VALUE OF THE RESET BUTTON

Jakob Nielsen of Useit.com, a Web site usability specialist, questions the value of the Reset button (http://useit.com/alertbox/20000416.html). He is not saying that the Reset button does not do what it is supposed to do, but rather that users who do not want to complete the form, rather than resetting it, usually click a link or type a URL and move on to another page. While a generalization, this does raise the question of the value of the Reset button in some forms. For example, basic contact forms found on many Web sites may not need a Reset button. Visitors who change their minds about submitting a message are likely to just move on. Armed with this knowledge, you may want to rethink whether you need a Reset button on every form.

Remember also that you may want to provide a button that clears data from a database, such as Clear This Order. While it may seem similar in function to a Reset button, the Clear This Order button may actually provide an action that should be processed by the form-processing application. For that reason, you should avoid button labels that could be confused with the functionality of the Reset button.

LESSON 5 Creating Radio Button Form Controls

The Radio button form control is designed to provide users with mutually exclusive options, such as male or female. It is possible, however, to improperly create Radio buttons so that they are not exclusive, for example, with the option "select three of your favorite sports." Because historically Radio buttons have always been used to allow users to select just one option from a group, breaking this pattern risks alienating your visitors. A similar option, the checkbox, is meant to provide multiple options, any or all of which may be selected. Just as with the text, password, and button form controls, you create Radio buttons using the `<input />` tag using the `type` attribute to identify the form control — `type="radio"`.

Radio buttons are commonly designed to work in a group, presenting options or answers to a common question. The individual Radio buttons are related to each other through a common `name` attribute. For example, `name="gender"` would be common to both male and female Radio buttons, and `value="male"` and `value="female"` would distinguish the two. By having a common `name` attribute, only one option from a group of Radio buttons may be selected. However, if the `name` attribute for the female option was set to gender and the `name` attribute for the male option was set to sex, the two options would not be related, making it possible to select both male and female, which, of course, is not a possible option.

The **value** attribute of the text and password controls (applied using the Init Val field in the Property inspector) inserts initial or default text in the field. The Radio button also has an Initial state option in the Property inspector, offering Checked and Unchecked options. Selecting the Checked option inserts the **checked** attribute in the **<input />** tag. The **checked** attribute is an unusual attribute in that there is only one attribute value — checked. If Checked is selected for the example of gender above, the **<input />** tag is written as **<input type="radio" name="gender" value="female" checked="checked" />**. Although HTML 4.01 and earlier versions allowed empty attribute values such as just **checked**, in XHTML, all attributes must have attribute values: checked must be entered as **checked="checked"**. If a Radio button is not to be checked, then the checked attribute is not included in the **<input />** tag at all.

Labeling a Radio button group is more complex than password or text fields. Just like password and text fields, each Radio button requires its own label so that you know if you are selecting male or female. However, for greatest accessibility, the group of related Radio buttons also should be labeled so that it is clear that the buttons belong to the same group. This is achieved using the **<fieldset>** and **<legend>** tags; these tags will be covered in *Lesson 9: Grouping Form Controls and Options*.

The Input Tag Accessibility Attributes dialog box allows you to select whether the label text should be positioned before or after the form control. By default, the dialog box presets the Before position for text and password fields and presets the After position for Radio buttons. Although using labels after Radio buttons is common practice, some accessibility advocates recommend that the label should precede the button so that the screen-reader user does not have to go back to the button to select it.

Dreamweaver provides two methods of creating Radio buttons: individually or in a group. In this lesson you learn both methods. Although using the Radio group method is easier, there will be times when you may need to add more buttons after you have created a group. You also learn why you must use a common **name** attribute for all Radio button options.

Create Individual Radio Buttons

1 **Open registration.html.**

2 **Click to the right of the Full Name text field and press Enter/Return. Type `Select your gender:` and press the Spacebar.**

We are not using the **<fieldset>** and **<legend>** tags to label this group of Radio buttons; rather, we precede the buttons with text that should make it clear what is being asked of the visitor.

3 **From the Forms Insert bar, select the Radio Button button.**

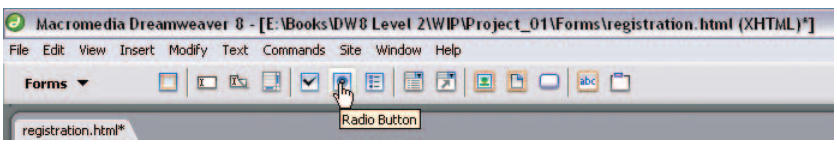

FIGURE 1.17

4 **Type `Male` in the Label field and click OK.**

Note that the Position has been preset to After form item.

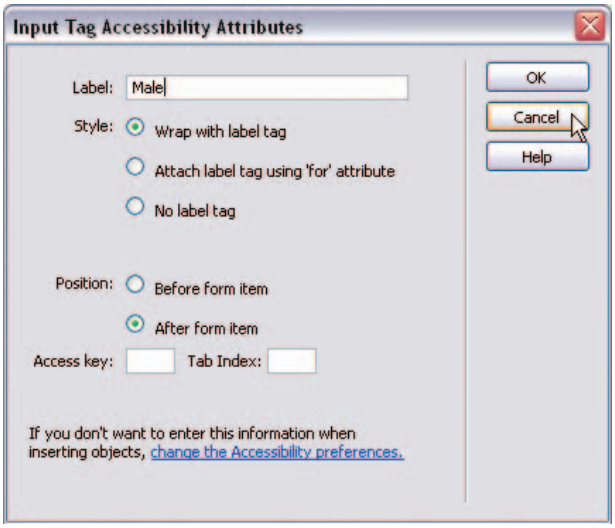

FIGURE 1.18

5 **Select the Radio button in the Document window. In the Property inspector, type `gender` in the Radio Button name field, type `male` in the Checked Value field, and leave the rest of the options at their defaults.**

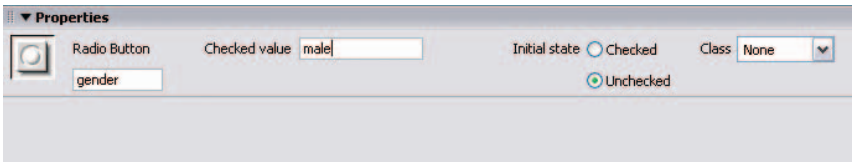

FIGURE 1.19

6 **Insert another Radio button to the right of Male with the label `Female`. In the Property inspector, type `sex` in the Radio Button name field and `female` in the Checked Value field.**

7 **Preview the page in your browser and select both Male and Female.**

FIGURE 1.20

8 **Return to Dreamweaver without closing your browser, change the name attribute of the Female Radio button to `gender`, and preview and test the page in your browser again.**

Now that both Male and Female Radio buttons share the same name attribute, only one of the two may be selected.

9 **Return to Dreamweaver, leaving the file open for the next exercise.**

Create Multiple Radio Buttons

In this exercise, you learn to create a group of Radio buttons using the Radio Group button from the Forms Insert bar. This function has a few beneficial features: it creates proper labels for the Radio buttons, it applies the same name attribute for all Radio buttons in the group, and it gives you the option of formatting the Radio buttons with line breaks or table cells.

1 **Click to the right of the Street Address text field and press Enter/Return.**

2 **Type `Select your dwelling type:` and press the Spacebar.**

3 **Click the Radio Group button in the Forms Insert bar.**

4 **In the Radio Group dialog box, type `dwelling` in the Name field. Under Label, click Radio and type `Apartment`. Press Tab and type `apt` in the Value column and press Tab. Type `Row House` and press Tab. Type `rhouse`.**

5 **Click the plus symbol (+) above Label to create a new row. Type `Individual House` and press Tab. Type `ihouse`.**

6 **Leave the Lay Out Using option set to Line Breaks and click OK.**

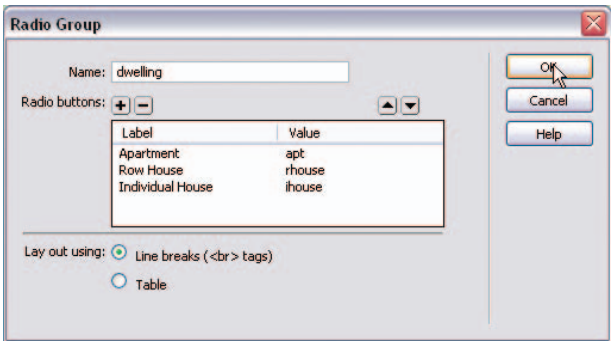

FIGURE 1.21

7 **To remove the line breaks, move the insertion point to the Apartment line, press the End key, and then press Delete/Forward Delete. Press End and then Delete/Forward Delete twice more to bring all of the Radio buttons onto the same line.**

8 **Preview the page in your browser and test the Radio buttons.**

The Radio Group dialog box assigns the same name attribute to all Radio buttons, ensuring that only one of the options can be selected.

9 **Return to Dreamweaver, leaving registration.html open for the next exercise.**

Add Another Radio Button to the Group

In this exercise, the option Condominium was forgotten and must be added to the group of Radio buttons of dwelling types. You learn that the Radio Group dialog box is useful for creating a group of Radio buttons but does not allow you to edit or add to the group. Instead, you must use the previously learned technique to add more Radio buttons to a group.

1 **In registration.html, click anywhere in the group of Radio buttons of dwelling types.**

2 **Click the Radio Group button from the Forms Insert bar.**

Notice that the settings are the same as the original default settings. This function does not pick up on the settings of the Radio buttons in which the insertion point is positioned. This dialog box is only useful for creating a new group of Radio buttons.

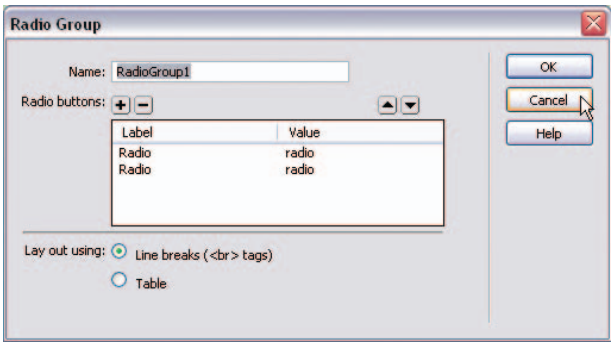

FIGURE 1.22

3 Click Cancel to close the Radio Group dialog box.

4 Click the Radio button (not the label) of any of the three dwelling options and note the Radio Button name.

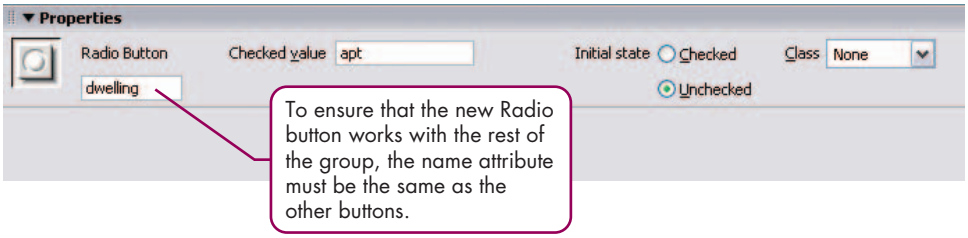

FIGURE 1.23

5 Click to the right of Individual House and press Spacebar.

6 Insert a new Radio button (not a group, just an individual button). Assign it the label of `Condominium`. In the Property inspector, assign it the Radio Button name `dwelling` and the Checked Value `condo`.

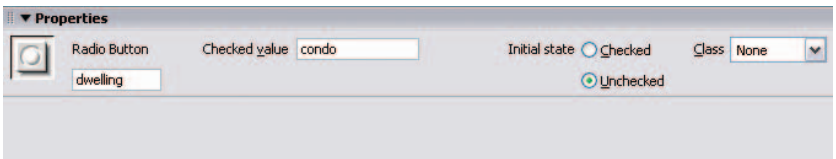

FIGURE 1.24

7 Preview the page in your browser to ensure that the new Radio button works with the rest of the group.

8 Close your browser and return to Dreamweaver, leaving registration.html open for the next exercise.

To Extend Your Knowledge . . .

THE OTHER OPTION

It is fairly common to see forms with several options for a particular question, and the last option is Other with a text field to its right, expecting you to complete the Other text field if Other applies to you. This creates a difficult situation for the developer of the form-processing application — one that you should avoid whenever possible. For example, the Radio button group might ask the visitor's profession and list options of Firefighter, Police, Retail, Management, and Other. If the listed professions were not applicable to most users of the form, more users would select Other than the identified professions. Among those who select Other, one person might type Actor in the Other text field where another might type Actress, listing essentially the same profession in two different ways (and spelling mistakes could create the same situation). Further, some might feel that they are in management in their government job and select Management, adding Government to the Other text field as well.

In all of these examples, it is virtually impossible to create a form-processing application that will handle this type of data easily. These types of situations require someone to review all entries to make sense of the data. On the other hand, it may be that Other is the best means to categorize data that does not fit within any of the listed options. Evaluate the form and try to anticipate the type of data you expect to collect before publishing the form on the Web site for visitors to complete.

LESSON 6 Working with Multiple-Choice Checkboxes

The checkbox form control shares the `<input />` tag with all of the other form controls discussed previously and, similarly, is differentiated from them using the `type` attribute, as in `type="checkbox"`. Checkboxes are similar to Radio buttons in that a group of checkboxes pertaining to the same question should all have the same `name` attribute; the different options presented by the checkboxes are specified in their `value` attributes. Checkboxes also have the Checked/Unchecked options, just as Radio buttons do. However, unlike Radio buttons, checkboxes are designed to allow multiple selections from a group.

Dreamweaver allows you to add Radio buttons individually or as a group, but the group option is not available for checkboxes: you must create them individually. Therefore, you must remember to use the same `name` attribute so that the options relate to the same question or topic.

Create Checkboxes

1 **In registration.html, click to the right of Condominium, press then End key, and then press Enter/Return.**

2 **Type** `What type of plants are you looking to purchase?` **and press the Spacebar.**

3 Click the Checkbox button from the Forms Insert bar.

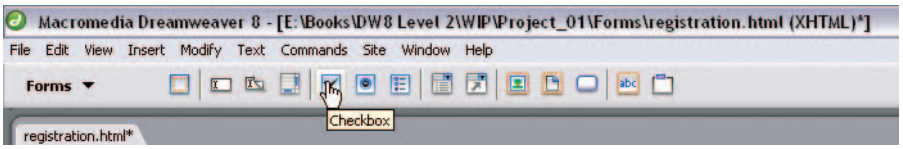

FIGURE 1.25

4 Assign `Indoor` as the label.

5 Select the Checkbox button and, in the Property inspector, type `plant_type` in the Checkbox Name field, type `indoor` in the Value field, and select Checked from the Initial state options.

6 Click the Checkbox button from the Forms Insert bar and assign `Outdoor` as the label.

7 Create another checkbox with the Checkbox Name field set to `plant_type` and the Value field set to `outdoor`. Leave Initial State set to Unchecked.

8 Click to the right of Outdoor, press Enter/Return, and insert a Submit button.

9 Save the page, leaving it open for the next exercise.

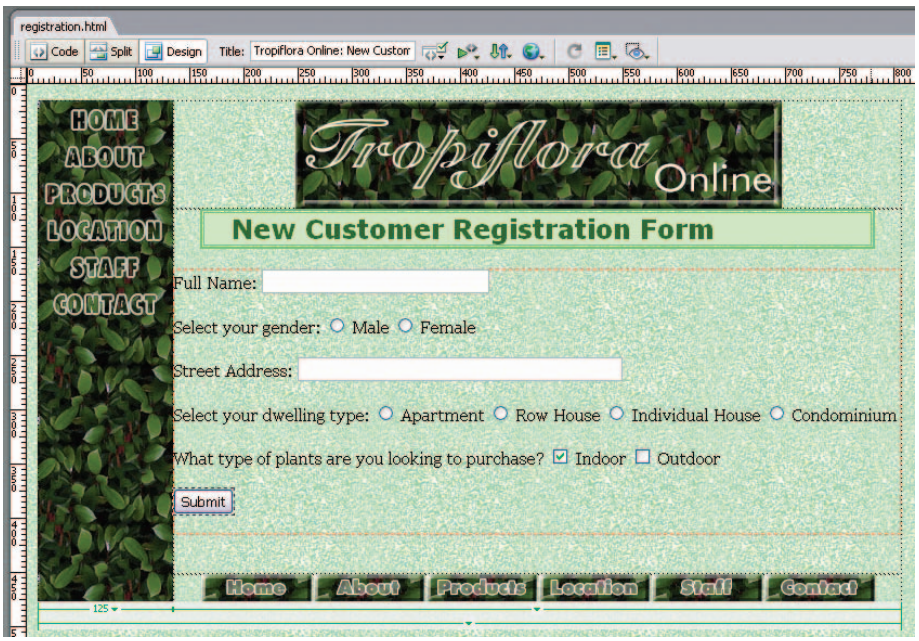

FIGURE 1.26

Compare Radio Button and Checkbox Query-String Data

In this exercise, you learn how the Radio button and checkbox data is sent to the form-processing application. Only one option from a Radio button group may be selected; therefore, there will be only one **`name=value`** pair in the query string for the Radio button. However, checkboxes allow multiple selections, so for each selected item from a checkbox group, there is a corresponding **`name=value`** pair.

1 **Preview the open registration.html page in your browser.**

2 **Select your gender, leave Indoor selected, and click the Submit button.**

3 **Click in the address bar of your browser, press the End key, and examine the query string.**

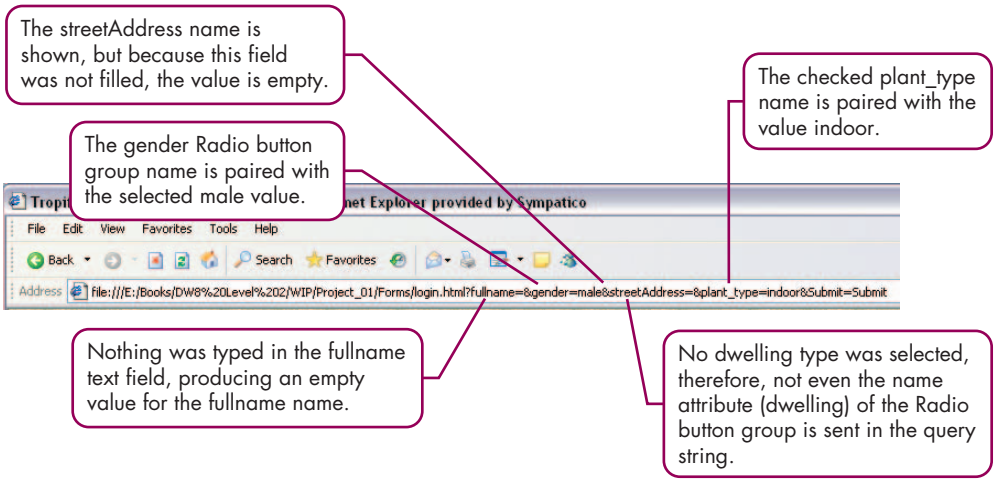

FIGURE 1.27

4 **Click the Back button in your browser.**

5 **Select a different gender, select a dwelling type, select both Indoor and Outdoor plant types, and click Submit. Examine the query string again.**

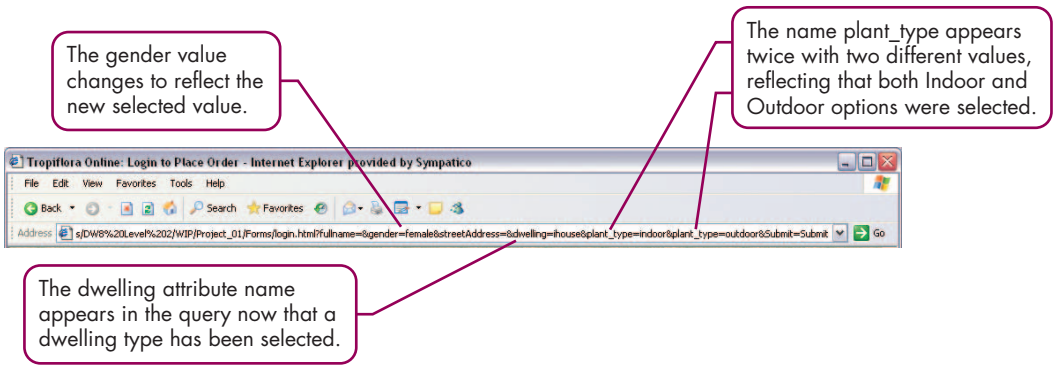

FIGURE 1.28

6 **Close your browser, return to Dreamweaver, and close registration.html.**

To Extend Your Knowledge . . .

PROCESSING RADIO BUTTON AND CHECKBOX OPTIONS

Processing Radio button options is not very difficult: only one option is selected for each group. Processing checkbox options is more difficult in that you must check for multiple instances of the same name, such as `plant_type=indoor&plant_type=outdoor`.

The usefulness of the checked option for Radio buttons and checkboxes may not be clear. As you saw in the previous exercise, if none of the buttons for a Radio button group are selected, the name of the group is not passed in the query string. The same principle applies to checkboxes, too. On a purchase order form, if there are shipping options for a product, one of them must be selected or the order cannot be shipped. The Web-site designer may choose to preselect the least-expensive option, leaving it to the user to accept or change the option.

Another use for preselecting an option is when information has already been entered and the user is asked to modify or confirm the information. For example, an e-commerce application remembers (by requesting the data from its database) that Bob used his MasterCard to pay for his last purchase. The e-commerce application may create the Payment Method form and preselect the MasterCard option for him; Bob is then free to accept or change the preselected payment option.

While you might not be willing to preselect (assume) Male or Female as the gender of a new customer, there are other circumstances when preselecting an option improves the usability of the form for your visitors.

LESSON 7 Labeling Form Controls

There are two aspects to labeling forms, one obvious, the other not. Many Web designers who are not knowledgeable of Web accessibility may place text beside a form control, such as a text field or Radio button. It might seem logical to assume that visitors would deduce the purpose of the control from the proximity of the text to the control. However, this relationship is not as clear to people who use screen readers. Using the `<label>` tag most certainly clarifies this situation for users of screen-reader software.

Using the `<label>` tag also improves the usability of Radio buttons and checkboxes for everyone. Without a `<label>` tag, to select either a Radio button or checkbox option, you must click the Radio button or checkbox. If you use the `<label>` tag, however, the width of the selectable area expands to include the label text. The user may click the label text to select the option, not just the control itself. If you click a label associated with a text field, the insertion point appears within the field, and if you click the label of a Radio button, that option is selected.

There are two methods for applying the `<label>` tag: explicit and implicit. The `<label>` tag is a container tag that requires its closing `</label>` tag. The implicit method of applying the `<label>` tag is to enclose both the label text and the form control between the opening and closing `<label>` tags, as in `<label>Full Name: <input type="text" name="fullname"></label>`. The explicit method of applying the `<label>` tag separates the label text from the form control, but using the `for` and `id` attributes, the relationship between the label text and the form control is explicitly stated. To apply this properly, the form control must be assigned an `id`, and the `for` attribute of the `<label>` tag uses the same value as the `id` of the form control, not unlike the `usemap` attribute of the `<img />` tag and the `name` attribute of the `<map>` tag, which relate the two. The following is an example of the explicit method of the `<label>` tag: `<label for="fullname">Full Name:</label> <input id="fullname" name="fullname" type="text" />`.

Dreamweaver supports both methods using the Input Tag Accessibility Attributes dialog box, although, so far, you have used only the implicit method. Dreamweaver also automatically inserts the `<label>` tag when Radio buttons are created using the Radio Button dialog box (activated by the Radio Control button of the Forms Insert bar), but the Radio and Checkbox buttons of the Forms Insert bar depend on the Input Tag Accessibility Attributes dialog box to insert the `<label>` tag.

You can use tables to lay out forms, commonly with the labels in one column and the form controls in another column. You may have noticed that the Radio Button dialog box did provide a table layout option (we chose Line Breaks instead). When you use tables to lay out forms, by placing the label text in one cell and the form control in another cell, you cannot use the implicit label method because the `<label>` tag cannot cross the cell boundary. For this reason, you would need to use the explicit label method. However, Dreamweaver does not support the explicit method in the Design view. You must work in the Code view to apply the `for` attribute.

Create Implicit Labels

In this exercise, you create implicit labels to associate label text with the form controls. You learn that the Radio Button dialog box inserts the `<label>` tag when you create a Radio button group using that dialog box. You also see how the `<label>` tag improves the usability of Radio button and checkbox controls.

1 **Open e-mail.html page, click in the text label Owner, and notice that the `<label>` tag appears in the Tag selector.**

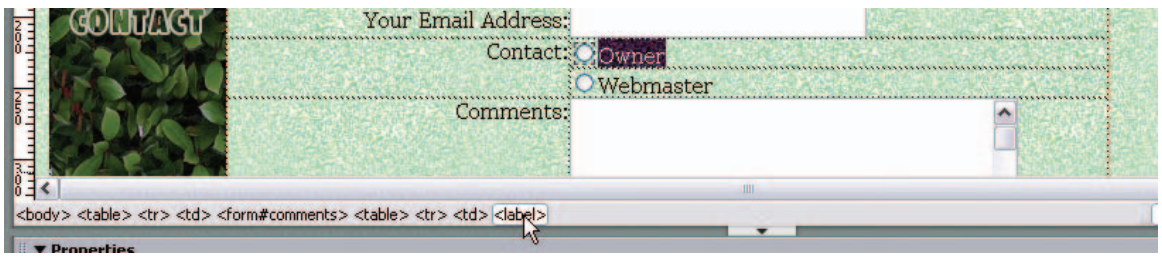

FIGURE 1.29

2 **Click Webmaster and notice that there is no `<label>` tag in the Tag selector.**

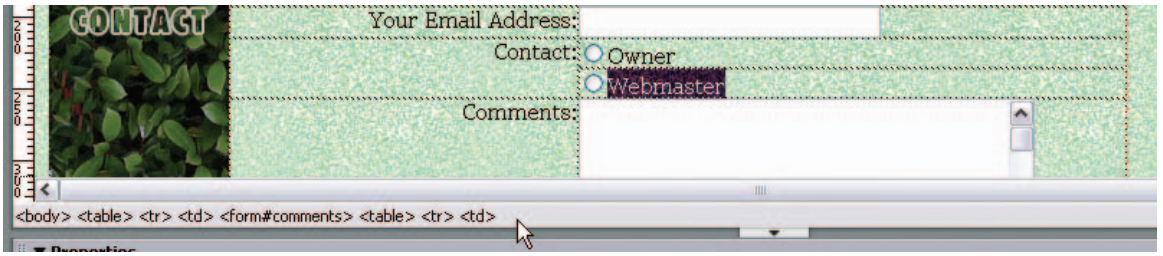

FIGURE 1.30

3 **Preview the page in your browser and click the text Owner.**

4 **Click the Webmaster text and then click the Radio button for Webmaster.**

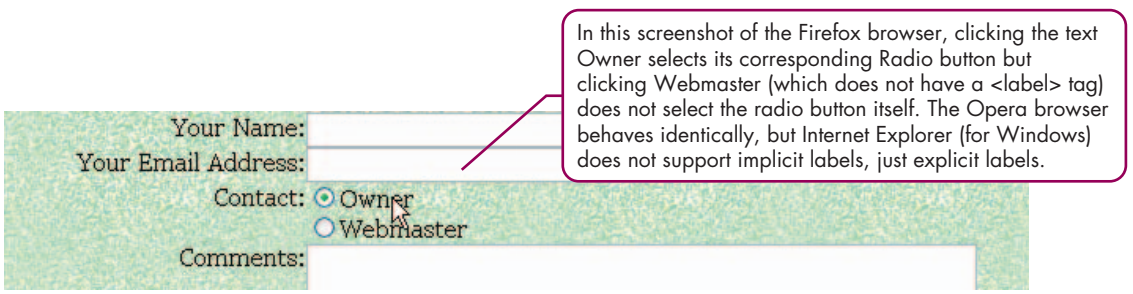

FIGURE 1.31

5 **Close your browser and return to Dreamweaver.**

6 **Drag to select Webmaster and its Radio button, and click the Label button from the Forms Insert bar.**

The Document window switches to the Split view.

7 **Click the Design button to close the Code window.**

8 **Preview the page in your browser.**

If You Have Problems

Internet Explorer for Windows does not support the implicit method of applying the `<label>` tag; therefore, if you do not have access to any browser other than Internet Explorer, you will not experience the usability benefit of applying the `<label>` tag to the label text for Radio buttons and checkboxes. If you have the opportunity to test this page with any of the Opera, Firefox, IE (Macintosh), or Safari (Macintosh) browsers, you will experience the benefit of implicit `<label>` tags to Radio buttons and checkboxes.

9 **Return to Dreamweaver but leave e-mail.html open for the next exercise.**

Create Explicit Labels

In this exercise, you learn to apply explicit `<label>` tags to the fullname and e-mail address text fields and the comments textarea field. To do so, you must assign `id` attributes to the male and female Radio buttons, apply the `<label>` tag to the Male and Female text labels, and add the `for` attribute to the `<label>` tags to explicitly relate the labels to their Radio buttons.

1 **In the open file, e-mail.html, Right/Control-click the text field for Your Name, and select Edit Tag <input> from the contextual menu.**

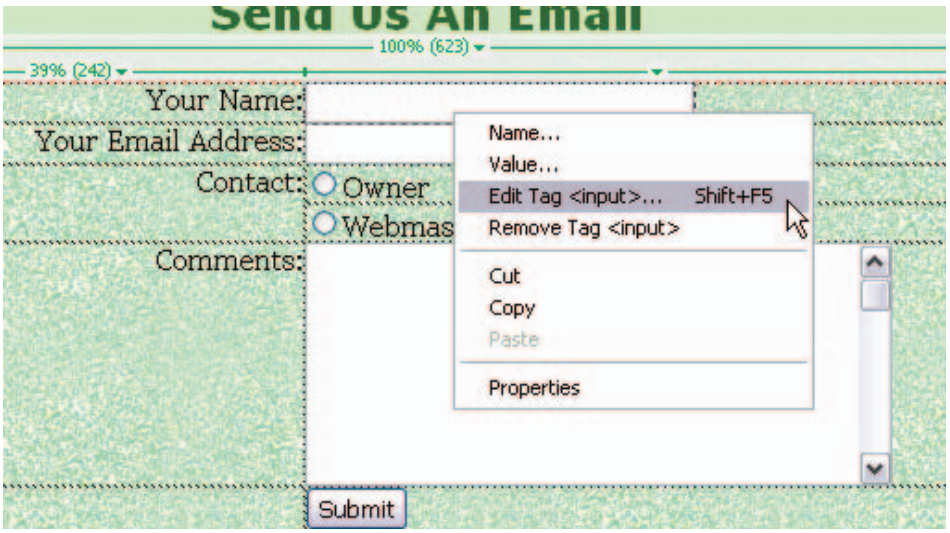

FIGURE 1.32

2 **In the Tag Editor–input dialog box, select the Style Sheet/Accessibility category, note that name appears in the ID field, and click OK.**

If the ID field were empty, you would need to type an ID into the field.

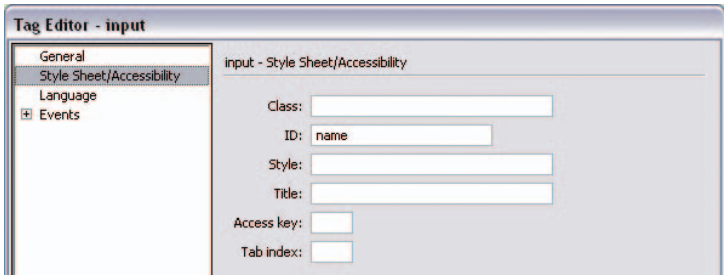

FIGURE 1.33

3 Select the text label Your Name: and Right/Control-click the selected text.

4 Choose Quick Tag Editor from the contextual menu.

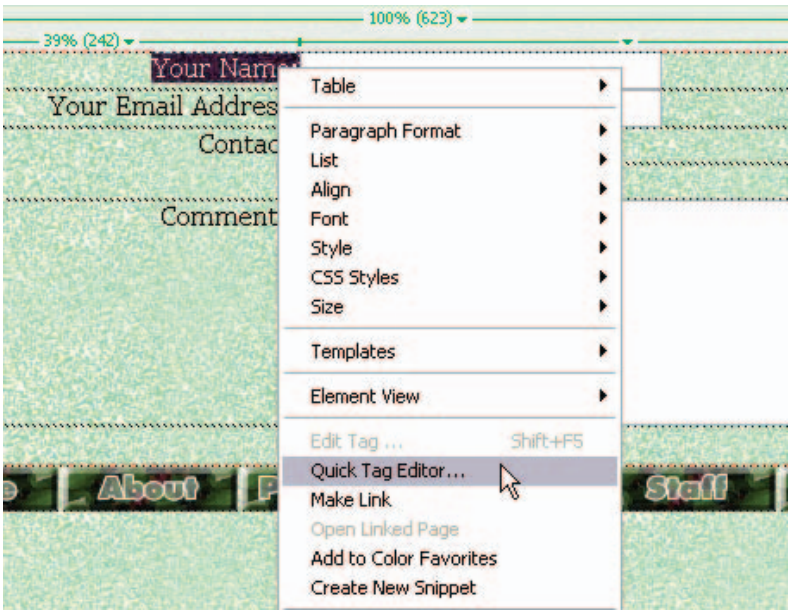

FIGURE 1.34

5 In the little editor window, press `l` (lowercase L) and press Enter/Return to accept `label`. Press the Spacebar, press `f`, and press Enter/Return to accept the `for` attribute. Type `name` and press Enter/Return.

FIGURE 1.35

6 **Repeat Steps 1 through 5, wrapping the text Your E-mail Address: with the `<label>` tag using the `for` attribute set to `e-mail`.**

7 **Repeat Steps 1 through 5, wrapping the text Comments: with the `<label>` tag using the `for` attribute set to `comments`.**

8 **Preview the page in your browser, click all of the text field labels, and note that the insertion point moves into the text field associated with the selected label.**

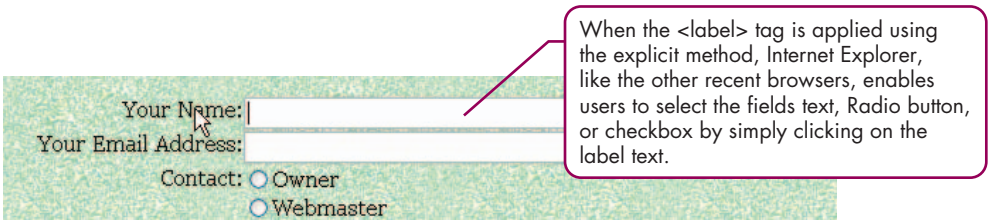

FIGURE 1.36

9 **Close your browser, return to Dreamweaver, and close e-mail.html.**

To Extend Your Knowledge . . .

INNOVATIVE USES FOR JAVASCRIPT IN FORMS

Forms often contain JavaScript functions, and there are many scripts that increase the usability of forms.

Text fields, as you know, can have initial text. Although initial text is useful, in some circumstances, to explain what users should enter into that field, it requires that users delete the initial text before they add their own. JavaScript can be written to delete the initial-value text when the user clicks or tabs into the field.

When shipping and billing addresses are the same, clicking a checkbox with the label Same as Billing Address can be used to trigger a JavaScript function that completes the shipping address fields with the same data as the billing address fields.

JavaScript can be written to provide options based on previous selections. For example, if USA is selected as the country, a list of the states appears, whereas if Canada is selected as the country, a list of the provinces and territories appears. Peter-Paul Koch, an eminent JavaScript developer, has taken this principle to the extreme. His contact form changes quite significantly depending on the purpose of sending him a message (http://www.quirksmode.org/contact.html). For example, if you select the subject Business from the list of possible subjects, a Radio button group appears with the options Nederland and Other.

Some forms contain both required and optional fields. Andy Clarke has created a form (with an accompanying tutorial) in which a JavaScript link hides the optional fields (http://stuffandnonsense.co.uk/archives/trimming_form_fields_3.html).

Finally, the granddaddy of all, validation: JavaScript is commonly used to validate the contents of a form to ensure that the e-mail address is formatted correctly, the date is valid, and all of the required fields have been completed. These are just a few of the options available for JavaScript validation.

Despite the apparent benefits for usability that these JavaScript functions provide, keep in mind that if accessibility is a concern, many of these JavaScript functions may not be accessible to everyone. Some forms are so dependent on JavaScript that if people were to try to use the form with an older browser or a JavaScript-incapable or disabled browser, they would be presented with an empty page. Despite the benefits that JavaScript may provide sighted users with current browsers, be aware of your site's audience and what limitations may result if you use JavaScript.

LESSON 8 Selecting From Lists

Select lists, named that because they are based on the **<select>** tag, can provide alternatives to both the Radio buttons and checkboxes: a select list can allow users to select just one item or multiple items. This is one of the few form controls that does not use the **<input>** tag.

There are two tags used to create select lists: the **<select>** tag, which, like the unordered list **<ul>** tag, surrounds the list of options, and the **<option>** tag, which, like the list item **<li>** tag, appears as often as there are items in the list. The **<select>** tag has three attributes of importance: the **size** attribute, the **name** attribute, and the **multiple** attribute.

The **name** attribute acts identically to the **name** attribute for other form controls: it identifies, to the form-processing application, the type of data being collected, such as shippingMethod or favoriteAnimal. The **size** attribute does not set the width of the control but, instead, the height in single units, such as 1, 2, or 3. When the **size** is 1 or unspecified (which defaults to 1), the select list acts as a pop-up list. However, if the **size** is 2 or more, the select list displays as a scrolling list. There is no attribute that may be used to specify the width of the **<select>** tag — by default, it stretches to fit the width of the widest item in the select list. However, the width may be specified using CSS. The **multiple** attribute, like the checked attribute of Radio buttons and checkboxes, has only one attribute value: **multiple="multiple"**. If **multiple** is inserted in the **<select>** tag, the user can select multiple items from the select list, whereas without it, the user can only select one item. Dreamweaver distinguishes a single select list from a multiple select list by naming them differently: a menu allows you to select just one option whereas a list allows you to select multiple options.

The **<option>** tag has only two attributes: the **value** attribute and the **selected** attribute. The opening and closing **<option>** tags enclose the option such as the carrot in a list of vegetables: **<option>**Carrot**</option>**.

The **value** attribute of the **<option>** tag is optional. For example, **<option>**Carrot**</option>** is equivalent to **<option value="Carrot">Carrot</option>**. In this example, the **value** attribute contains the same text that is visible in the list. At other times, especially if the list is populated from a database system, you may need to display a product description but assign the **value** attribute with the product number, such as **<option value="PN2318L">Black & Decker: Chainsaw</option>**.

In this example, although the users would see Black & Decker: Chainsaw, the value sent to the form-processing application would be PN2318L, the product number, and the form-processing application would act on this product number rather than the displayed text.

The `selected` attribute is similar to the `multiple` attribute of the `<select>` tag; it has only one attribute value: **selected="selected".** The `<option>` tag with the `selected` attribute displays in the list, whether it is the first, middle, or last option in the list. If the `<select>` tag has the `multiple` attribute, multiple `<option>` tags may have the `selected` attribute although only the first will be displayed unless the `size` attribute is set large enough to display two or more of the preselected options.

Create a Menu

In this exercise, you create a select list from which only one option may be selected. You set one item to be initially selected and learn that you may go back and edit an existing select list.

1 **Open registration.html, click to the right of Outdoor, and press Enter/Return.**

2 **Click the List/Menu button on the Forms Insert bar.**

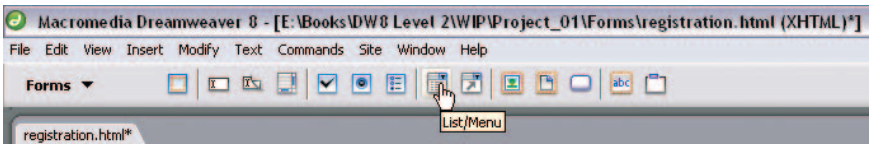

FIGURE 1.37

3 **In the Label field, type `How did you hear about Tropiflora Online?`, ensure that Style is set to Wrap with label tag and Position is set to Before form item, and press Enter/Return.**

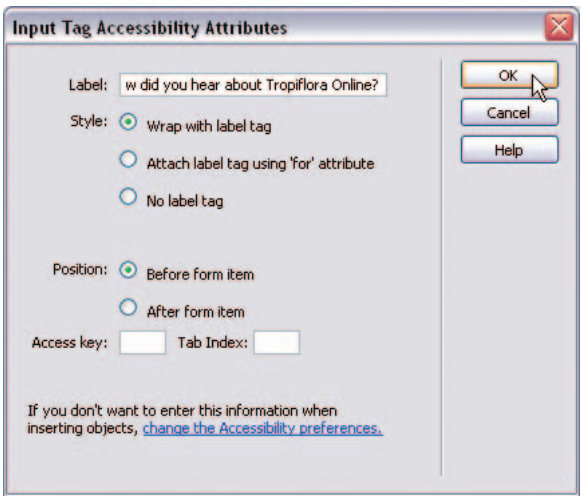

FIGURE 1.38

4 **Type `hear` in the List/Menu name field, and then click the List Values button.**

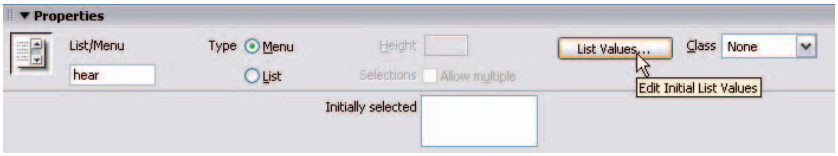

FIGURE 1.39

5 **Under Item Label, type `Magazine`, press Tab twice, type `Search Engine`, press Tab twice, and continue to create the following list (the last item is `Got lost on Tallevast Road`). Click OK when finished.**

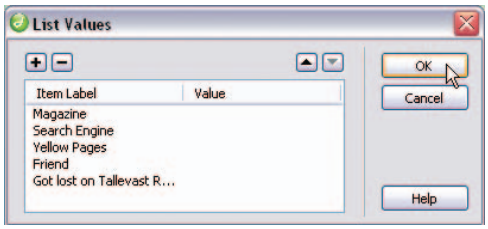

FIGURE 1.40

6 **Click `Magazine` in the Initially Selected field, and notice that Magazine displays in the list in the Document window.**

7 **Control/Command-click Magazine to deselect it, and then click Yellow Pages.**

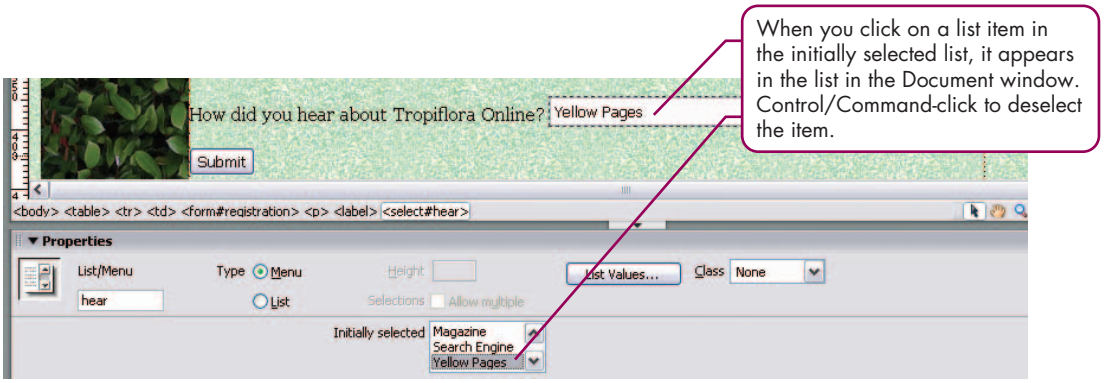

FIGURE 1.41

8 **Click the List Values button in the Property inspector.**

Notice that unlike the Radio Button dialog box, in which you cannot edit a group of Radio buttons, the List Values dialog box is filled with the existing options, allowing you to edit, delete, or add to the list.

9 **Complete the Value column as shown in the following graphic and click OK.**

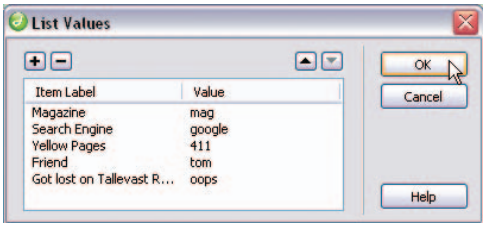

FIGURE 1.42

10 **Preview the page in your browser.**

Notice that the browser window provides no evidence that the option values are different from the option labels. However, a form-processing application only uses the option values.

FIGURE 1.43

11 **Click the Submit button and examine the query string in your browser's address bar.**

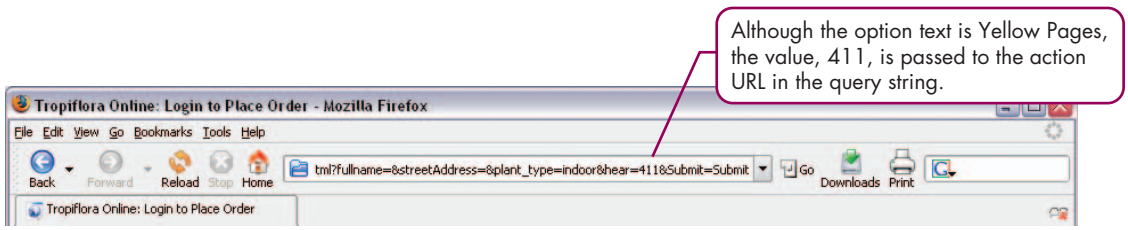

FIGURE 1.44

12 **Close your browser and return to Dreamweaver, leaving registration.html open for the next exercise.**

To Extend Your Knowledge . . .

ADDING INSTRUCTIONS TO SELECT LISTS

Many Web designers add another option to the top of a select list. This option contains information or instructions on how to use the list, such as "Select your year of birth," "Choose a shipping method," or just "Country." We recommend that, in collaboration with the developer of the form-processing application, you create a dummy value for the option, enabling the developer to watch for it in the program. If this is a required field of the form, the program must send the form back to the user to complete this option. For example, in a list of countries, the first option with the Select Your Country label may be coded as follows: `<option value="noCountry" selected= "selected">Select Your Country</option>`, and the Web developer would make the program watch for a value of noCountry.

Create a List

Dreamweaver calls the multiple select option list a *list*. In fact, the only difference between Dreamweaver's list and menu in code is that the `<select>` tag of the list contains the `multiple` attribute whereas the menu does not. Furthermore, the ability to apply the `size` attribute to the menu is not available from the Property inspector. What would you do to create a single select menu that is two or more lines in height? You would apply `size="2"` (or more) directly to the code either using the Code window or Right/Control-clicking the List/Menu control in the Design window and selecting Edit Tag <select>.

In this exercise, you create a list from which multiple items may be selected. You also preselect multiple items using Control/Command-click.

1 **In the open file, registration.html, click to the right of the select list and press Enter/Return.**

2 **Click the List/Menu button on the Forms Insert bar.**

3 **Type `Select the type of equipment you are interested in` in the Label field and click OK.**

4 **Click to select the list in the Document window and, in the Property inspector, type `equipment` in the List/Menu name field in the Property inspector, and select List from the Type options.**

5 **Set the Height to `5` and enable the Allow Multiple option.**

6 **Click the List Values button, enter `Gloves`, `Tools`, `Chemicals`, `Clothing`, and `Books` in the Item Label column, and click OK.**

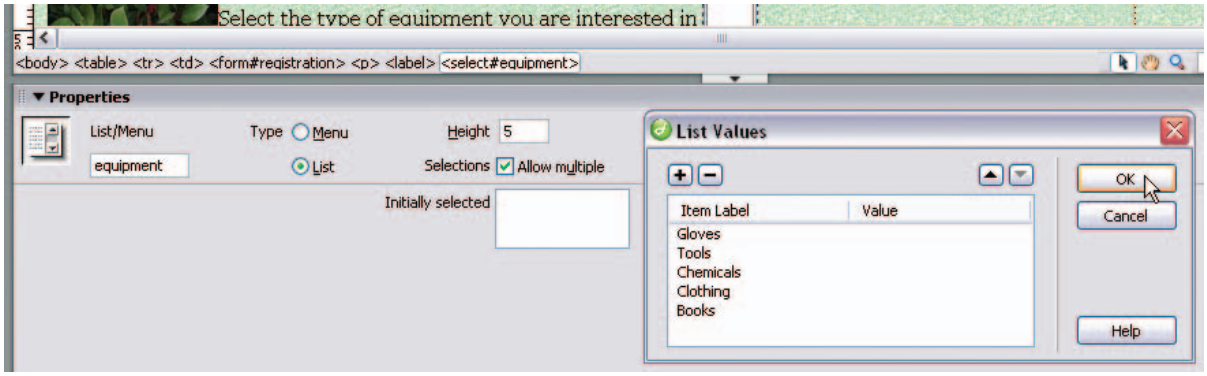

FIGURE 1.45

7 **Using the Initially Selected list, click Gloves and then Control/Command-click Chemicals.**

8 **Preview the page in your browser.**

Notice that both Gloves and Chemicals are selected.

9 **Click Tools, Control/Command-click Books, click the Submit button, and examine the query string.**

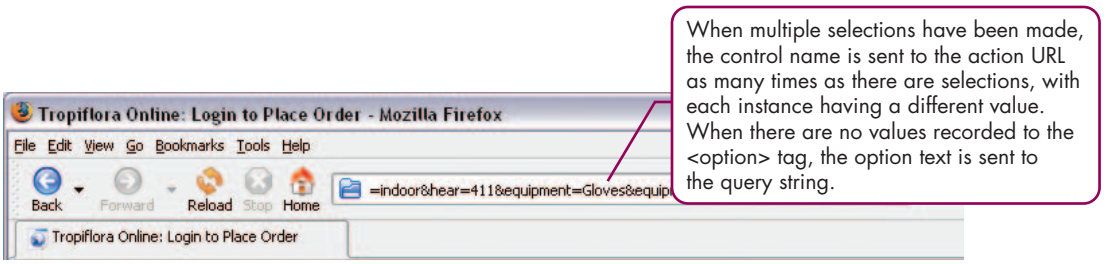

FIGURE 1.46

10 **Close your browser, return to Dreamweaver, and close registration.html.**

To Extend Your Knowledge . . .

MULTIPLE SELECT LISTS ARE UNCOMMON

It is uncommon to encounter multiple select lists. Often, when they are used, the Web page designer adds instructional text, such as "Control-click (Windows) or Command-click (Macintosh) to select multiple items." The other option is to avoid confusion altogether. Because multiple select lists are similar to checkboxes, which are more common and understood by a greater portion of the population, unless there is a compelling reason to use multiple select lists, you might consider using checkboxes instead.

LESSON 9 Grouping Form Controls and Options

You have learned that the `<label>` tag benefits all users by extending the selectable area for Checkbox(es) and Radio buttons to include the area taken by the text labels. You learned that the `<label>` tag benefits screen-reader users by relating the label text to the form control. There are two other form options that provide similar benefits — the `<fieldset>/<legend>` pair of tags and the `<optgroup>` tag.

The `<fieldset>` tag is used to collect a group of controls. For example, a series of text fields, Radio buttons, and checkboxes may be used to allow users to fill out the billing address for their orders. A virtually identical series of controls may be used to allow users to fill out the shipping address for their orders. To reduce the possibility of confusion, the billing address controls may be enclosed in one `<fieldset>` tag and the shipping address controls may be enclosed in another `<fieldset>` tag, thereby creating, in code, anyway, two groups of controls that are distinctly separated. The `<legend>` tag also plays a role: any text between the opening and closing `<legend>` tags appears above the group, thereby labeling the group. The `<fieldset>` tag also has a default presentation of creating a border around the group of controls enclosed within it.

The `<optgroup>` tag is used to group select-list options and assign names to groups. For example, UPS and FedEx both provide Regular and Express shipping services. Rather than list the four items as UPS-Regular, UPS-Express, FedEx-Regular, and FedEx-Express, you may group the UPS shipping methods in one `<optgroup>` and the FedEx shipping methods in another `<optgroup>`. The `label` attribute of the `<optgroup>` tag identifies UPS or FedEx, but these group names are not selectable.

Use the <fieldset> Tag to Group Controls

In this exercise, you surround a group of controls with the `<fieldset>` tag and label the group with the `<legend>` tag. You create three fieldset groups: the shipping address, the billing address, and the preferred shipper and method. You preview the page in your browser to view the default presentation of the `<fieldset>` and `<legend>` tags. Dreamweaver does not display the border that surrounds contents enclosed in the `<fieldset>` tag.

1 **Open addresses.html from the Forms site.**

FIGURE 1.47

2 Click in the first table of the form, and then click the `<table>` tag to the right of the `<form>` tag in the Tag selector to select the table.

3 Click the Fieldset button (last button) on the Forms Insert bar.

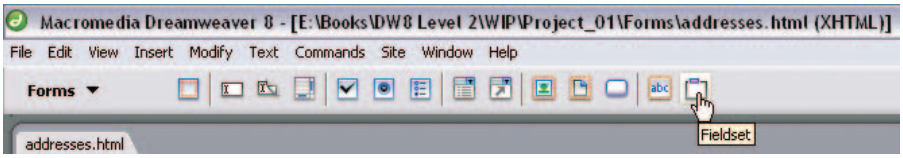

FIGURE 1.48

4 Type `Shipping Address` in the Legend field of the Fieldset dialog box and click OK.

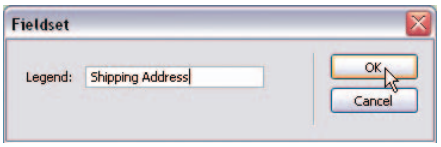

FIGURE 1.49

5 Click to the right of the selected table to deselect it.

6 Scroll down in the Design window. Using the same procedures as in Steps 2 through 4, select the second table, click the Fieldset button, and assign `Billing Address` as its legend.

7 Deselect the second table. Select the third table (consisting of just two cells), click the Fieldset button, and assign `Shipping Preferences` as its legend.

8 Preview the page in your browser.

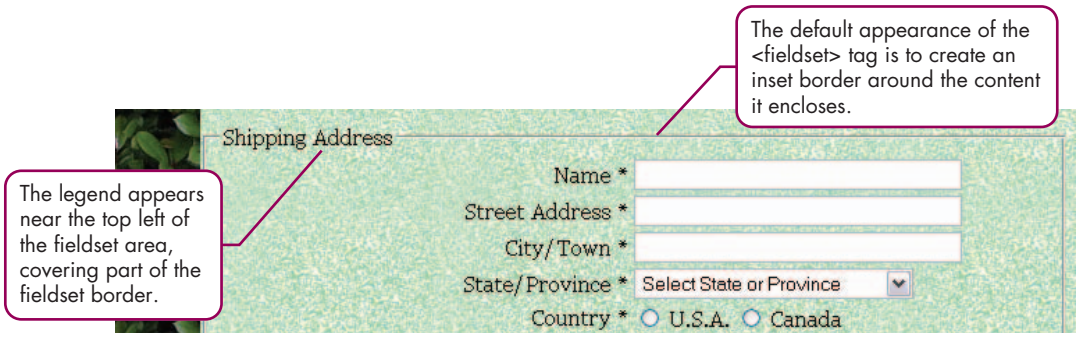

FIGURE 1.50

9 Return to Dreamweaver, leaving addresses.html open for the next exercise.

Create Option Groups in a Select List

In this exercise, we use the `<optgroup>` tag to create groups of options. This increases the usability of select lists, especially if they are long lists. The `<optgroup>` tag adds an additional item to a list, or so it appears. When you assign the `<optgroup>` tag to a group of select list options, you are prompted to name the group. The name is then applied to the `label` attribute of the `<optgroup>` tag. This name appears in the select list, but it is not selectable: it is like a heading for a block of options.

You cannot apply the `<optgroup>` tag in the Design window. You must work in the Code window.

1 **In the open file, addresses.html, select the first select list of States and Provinces and switch to the Code view.**

2 **Click to the left of `<option>`Alabama, scroll down to Wyoming, and Shift-click to the right of the Wyoming closing `</option>` tag to select all states of the United States.**

3 **Right/Control-click in the selected region, and select Insert Tag from the contextual menu.**

FIGURE 1.51

4 **Select HTML tags>Forms>optgroup, and click Insert.**

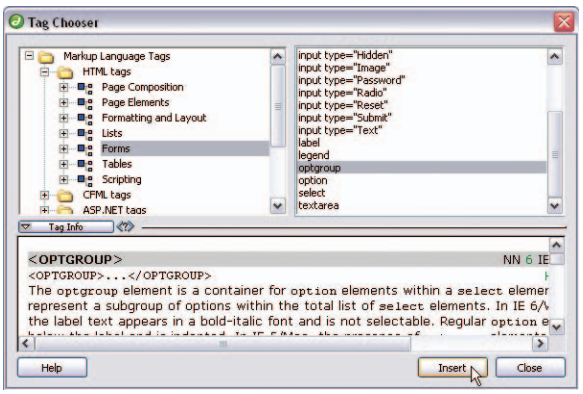

FIGURE 1.52

5 Type `US States` in the Label field, click OK, and then click Close to close the Tag Chooser dialog box.

FIGURE 1.53

6 In the Code window, click to the left of `<option>`Alberta, Shift-click to the right of the Yukon Territories, closing `</option>` tag, and insert the `<optgroup>` tag using `Canadian Provinces` as the Label. Close the Tag Chooser dialog box and the Code window when finished.

7 Apply Steps 1 through 6 to the second list of states and provinces.

8 Select the third select list of Shipping Preferences.

9 Select the two FedEx options (click to the left of the first `<option>` tag and Shift-click to the right of the second closing `</option>` tag), and insert the `<optgroup>` tag with the Label `FedEx`. Using the same procedures, assign `UPS` to the `label` attribute of the `<optgroup>` tag of the two UPS options and `USPS` to the two USPS options.

10 Given that the `label` attribute of the new `<optgroup>` tags identifies the FedEx, UPS, and USPS groups, delete `FedEx`, `UPS`, and `USPS` from the `<option>` label text (not the `<optgroup>` labels). For example, FedEx-Regular becomes just Regular.

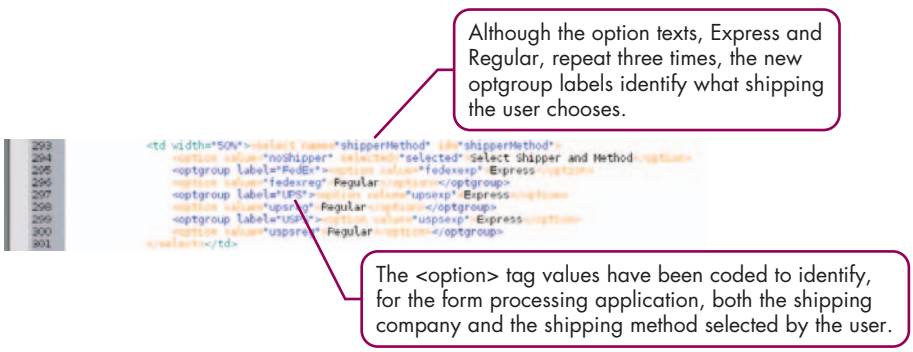

FIGURE 1.54

11 **Switch to the Design view, preview the page in your browser, and test the modified select lists.**

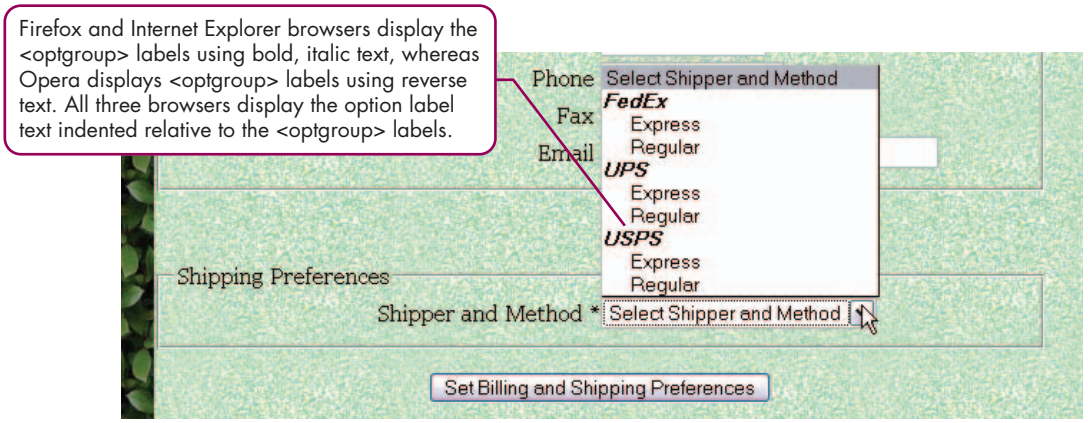

FIGURE 1.55

12 **Return to Dreamweaver, leaving addresses.html open for the next exercise.**

Modify the Style of <fieldset> and <legend> Tags

In this exercise, you modify the default style of the `<fieldset>` and `<legend>` tags. Legend text is blue in Internet Explorer and black in Firefox and Opera and is the size of paragraph text. To make the purpose of the group clearer to users, you change the color and size of the legend text. The `<fieldset>` tag, by default, creates a border around enclosed content: in Opera and Firefox, the border is a grooved style, whereas in Internet Explorer, it is a thin, flat, solid, gray borderline. You modify the borderline style of the fieldset and, to match the fieldset line, add a borderline to the legend.

1 **In the open file, addresses.html, Right/Control-click in the CSS Styles panel and select New. (If the CSS Styles panel is not open, first click the CSS panel-group title, click the CSS Styles tab, click the All button, and then Right/Control-click and select New.)**

2 **In the CSS Styles dialog box, set Tag as the Selector Type, type or select `legend` from the Tag list, ensure that Define in is set to This Document Only, and click OK.**

3 **In the Type category, set the Font to Verdana, Arial, Helvetica, sans-serif; Size to `1.2` ems; Weight to bold; and Color to `#336600`. Click OK.**

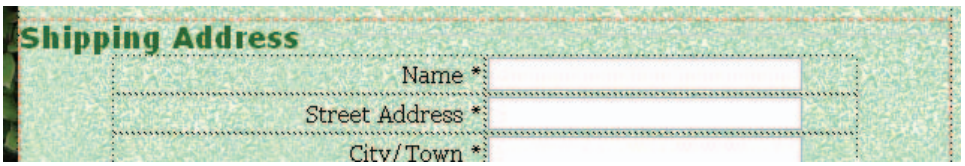

FIGURE 1.56

4 **Using the same procedure as in Step 2, create a new CSS style for the `fieldset` tag. From the Border category, leave Same for All checked, set Style to solid, Width to thick, and Color to `#336600`, and click OK.**

As you noticed in Step 11 of the previous exercise, Dreamweaver does not display the fieldset border in the Design window.

5 **To apply a similar border style to the legend text, Right/Control-click legend in the CSS styles panel and select Edit. In the Border category, leave Same for All checked, and set Style to solid, Width to thick, and Color to `#336600`.**

6 **In the Box category, uncheck Same for All in the Padding group, and set Top to `3` pixels, Right to `10` pixels, Bottom to `3` pixels, and Left to `10` pixels. Leaving Same for All checked for the Margin group, set Top to `0` pixels and click OK.**

Did you notice that the Padding, Margin, Style, Width, and Color groups in the Box and Border categories of the CSS dialog box, respectively, were surrounded by the software equivalent to fieldsets and legends? Many features of Web design come from application design.

7 **Preview the page in your browser.**

8 **Return to Dreamweaver, leaving addresses.html open for the next exercise.**

To Extend Your Knowledge . . .

DO NOT REFER TO COLOR

One of the important practices of Web accessibility is to accommodate both the blind and the colorblind. Many Web designers, when thinking about Web accessibility, forget that although the colorblind can see, their eyes cannot distinguish some colors correctly. Some colors, such as red and green, can appear almost black. When designing forms, required fields are often colored to suggest to the user that those fields are required along with a statement, such as: "You must complete the fields shown in red." The color blind will have some degree of difficulty identifying the required fields and, of course, the blind will not be able to see the color at all. You must use some other method of identifying required fields, such as using an asterisk (a statement at the top of the form indicating the purpose of the asterisk is also useful). You may color the label text (and the instructions to match) as long as the color combination and contrast are usable by the colorblind; just do not use the color as a criterion.

LESSON 10 Creating Hidden Fields

Hidden fields, as the name suggests, are not visible in the browser window, but these fields do exist in code. They are commonly used to pass information from the form to the form-processing application. For example, when a customer orders from an e-commerce site, his or her customer number may be stored in a hidden field and passed to the e-commerce application via the query string.

A hidden field uses the `<input />` tag with the `type` attribute set to `hidden`. The `name` attribute may be used to identify the type of data carried by the `value` attribute, such as `name="customerID" value="123456"`. You may use as many hidden fields as necessary for your application, but understand that any sophisticated user can view the information in the hidden fields by simply looking at the code. Do not store a password or any other critical information in a hidden field. There are other more secure methods available, such as ***cookies*** (text that is passed between your browser and the Web server) and session variables. (***Session variables*** are text that identifies you and may be passed as a cookie, stored in links, or stored in hidden fields. The session-variable text is just an identification code that is validated by the Web-server application and does not carry any recognizable information.)

Create a Hidden Field

In this exercise, you create a hidden field that contains an encrypted session variable to identify who you are and what database record you are updating through the use of the form.

1 **In the open file, addresses.html, click in the empty paragraph between the Shipping and Billing fieldsets.**

2 **Click the Hidden Field button on the Forms Insert bar (third from the left).**

3 **In the Property inspector, type `sid` in the HiddenField Name field (sid is a common short name for session id), type `c8b068f108ccbb25e6b43efd2dd13c66` in the Value field, and press Enter/Return.**

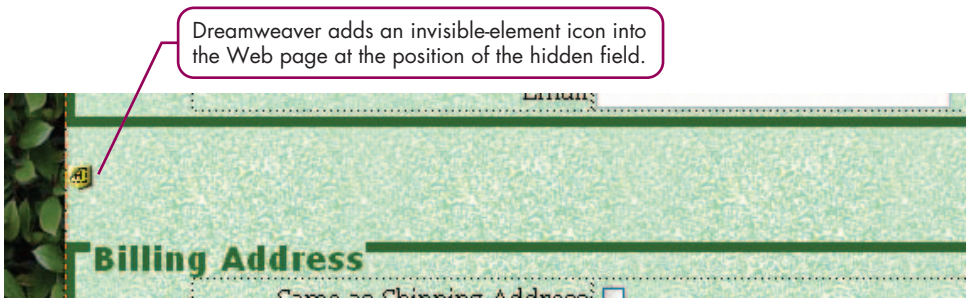

FIGURE 1.57

4 **Preview the page in your browser, and search the query string for the sid variable. (It is a long query string, so you will have to scroll to find it.)**

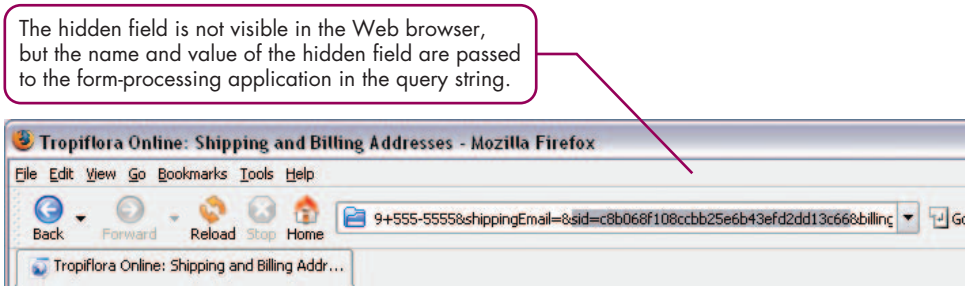

FIGURE 1.58

5 **Close your browser, return to Dreamweaver, and close addresses.html.**

To Extend Your Knowledge . . .

SPAM, FORMMAIL APPLICATIONS, AND HIDDEN FIELDS

With the exception of the mailto: action URL, form-processing applications must be installed and run on a Web server, which is not something a free Web-hosting service (whether through your ISP or a third-party service, such as Tripod.com) will allow. However, many free Web-hosting services do recognize that their clients may wish to create some basic forms, such as a contact form with the data collected from their visitors e-mailed to them. Rather than providing you direct access to the application, these hosting services often provide you the action URL and some instructions on how to configure the form to send your visitors' data to you via e-mail. This is not the same as a mailto: URL; the formmail application (an application that e-mails collected form data to the intended recipient) uses its own programming to e-mail the data to you.

If you do not have access to the formmail application, how does it know to e-mail your visitors' data to you? This is often done using a hidden field in which the name and value are set using `name="e-mail-address" value="you@domain.com".` The formmail application then takes the data collected by the form and e-mails it to the e-mail address in the value attribute.

In Chapter 6 of *Essential For Design: Dreamweaver 8 Level 1*, you learned that forms are a good way to hide your e-mail address from spammers, but not if you store the e-mail address in a hidden field. Hidden fields are no more secure than an e-mail link on a Web page. If you must hide your e-mail address, you must have access to the formmail application and store your e-mail address right in the application. Only someone with rights to access the files on your Web server (or your space in one) can extract your e-mail address from the programming code. However, this is hacking and beyond the ability of spammers and their e-mail-harvesting software, so you may be assured that storing your e-mail address in the programming code will protect it. If you cannot afford anything but free Web space, recognize that your e-mail address in a hidden form field may be harvested at some point. You may try another free hosting service; there are many.

CAREERS IN DESIGN

FORMS DESIGN AND USABILITY TESTING

You have learned to create static pages that, with the exception of links, have no features with which visitors may interact. Forms add a degree of interaction between the visitor and the Web page by allowing the visitor to enter information or make selections. Even without dynamic JavaScript effects, visitors know that by clicking the Submit button, some effect will occur, such as sending an e-mail, posting a question to a discussion forum, or purchasing a book. To ensure that your visitors can use the form as you would like them to use it, however, before you create the form, you must carefully evaluate the type of data that you want it to collect and the type of form controls that you might use.

Sometimes the best input method is not obvious. For example, credit-card numbers are often in the form of 1234 2345 3456 4567. With spaces, this credit-card number takes 19 characters. One possible issue is that people may not enter the spaces and may think, therefore, that they are expected to insert the three-digit security number as well; if they do so, the 19-digit number is invalid. To avoid this problem, do you divide the credit card number into four input fields each containing four digits? Doing so means that after each field, the visitor must press Tab to move to the next field but, at the same time, this format makes clearer to visitors that they only need 16 numbers. Some developers add JavaScript to this type of form so that after the fourth number is pressed, JavaScript moves the insertion point to the next field. If, however, a user looks at the credit card and not at the screen and presses Tab after the fourth number, JavaScript would already have moved the insertion point to the next field so the Tab would move one the insertion point one field further, leaving a field empty.

If you need to develop either a complex or a unique form, consider doing some usability testing in which people test your form and give their feedback. Usability testing can be expensive, but it does not have to be. You can gather a group of friends or coworkers and have them test your form and provide feedback, either as written notes or in a group discussion afterwards. A Web site that depends on information collected by its forms must ensure that its forms are not an impediment to the success of the Web site.

SUMMARY

In this chapter, you learned about forms and the many roles that forms play in Web applications. You explored form controls. You learned how to send form data to a form-processing application and the methods by which it may be sent. You learned how to insert single- and multi-line text fields and password fields. You added Submit and Reset buttons and learned how buttons may be used to trigger other actions, such as edit and delete. You explored the differences between Radio buttons and checkboxes. You created menus and lists and learned that they may be used in place of Radio buttons and checkboxes. You learned why it is beneficial to everyone to properly label form controls. You grouped form controls into fieldsets and grouped options in select

lists and learned how both improve the usability of forms. You learned about hidden fields and how they may be used in a Web application. You learned that although, as a designer, you may assemble and design a Web form, that Web form is the gateway to a Web application that requires a programmer to create it so it can capture, analyze, and use the data.

KEY TERMS

Application
Blog
C++
CGI
Checkbox
Cookies
E-commerce
Encrypted
Form controls
Formmail
HTTPS/S-HTTP
Menu
Perl
Query string
Radio button
Remotely hosted
Scripts
Session variables
Select list

CHECKING CONCEPTS AND TERMS

SCREEN ID

Identify the indicated areas from the list below:

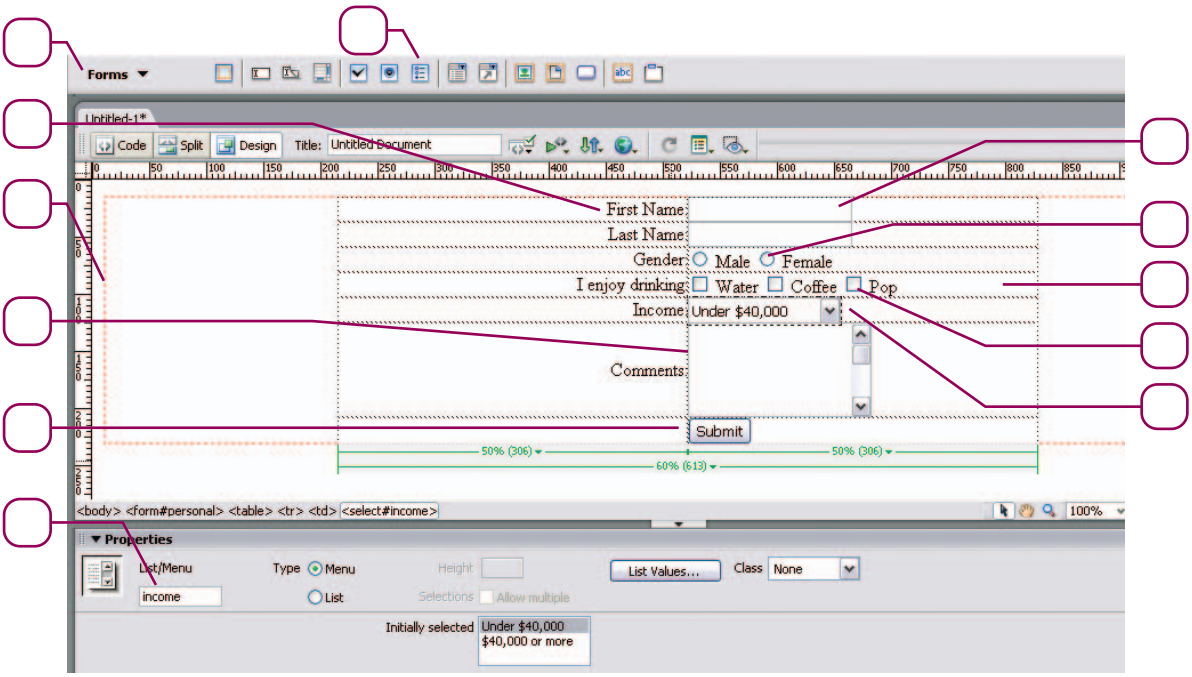

FIGURE 1.59

a. Forms Insert bar
b. Name of select list
c. Radio button
d. Label text
e. Radio group button
f. Text field
g. Checkbox
h. Button
i. Form outline
j. Layout table
k. Menu list
l. Textarea

MULTIPLE CHOICE

Circle the letter of the correct answer for each of the following:

1. Forms may be used to _____.
- **a.** pay bills online
- **b.** send an e-mail
- **c.** apply for a credit card
- **d.** All of the above

2. The two controls that may be used to replace each other are _____.
- **a.** Radio buttons and checkboxes
- **b.** checkboxes and multi-select lists
- **c.** legends and labels
- **d.** optgroups and fieldsets

3. If the action URL is search.php, the value is 3F9J, and the name is XV9C, the correct query string is _____.
- **a.** `search.php?value=3F9J&name=XV9C`
- **b.** `search.php=XV9C+3F9J`
- **c.** `search.php?XV9C=3F9J`
- **d.** `search.php/value=3F9J+name=XV9C`

4. What attribute would you use to display the third option in a select list?
- **a.** `display="3"`
- **b.** `selected="selected"`
- **c.** `checked="checked"`
- **d.** `multiple="multiple"`

5. Which button code would be best to display Modify and trigger an edit function?
- **a.** `<input type="edit" name="edit" />Modify`
- **b.** `<input type="submit" value="Modify" name="edit" />`
- **c.** `<input type="button" label="Modify" id="edit" />`
- **d.** `<input type="reset" name="Modify" action="edit" />`

6. Why does the W3C recommend labeling form controls?
- **a.** The colorblind have difficulty reading some text/background color combinations.
- **b.** Screen-reader software skips past unlabeled form controls.
- **c.** Labeled form controls improve the usability of the form.
- **d.** Label text displays in bold, increasing its readability.

7. In a select list of country names, which option would a Web application reject and request the user select a valid country?
- **a.** `<option>Select a country</option>`
- **b.** `<option value="none">Select a country</option>`

c. `<option value="notSelected" selected="selected">Select a country</option>`

d. All of the above

8. Which of the following is true?
   a. A group of Radio buttons should allow only one item to be selected.
   b. A group of checkboxes can allow multiple items to be selected.
   c. The multi-select list is uncommon and unfamiliar to most users.
   d. All of the above

9. The `<fieldset>` and `<legend>` tags may be used to identify _____.
   a. a group of Radio buttons
   b. a collection of a variety of form controls
   c. a group of related select-list options
   d. All of the above

10. The get method _____.
    a. sends the form data in the action URL
    b. encrypts the form data so a hacker cannot snoop it
    c. e-mails the form data to your e-mail address
    d. All of the above

DISCUSSION QUESTIONS

1. Online stores, from which you can purchase items, require a large amount of information to enable you to complete your purchase. What type of information would you need to collect from customers to enable them to complete their purchases? For each type of information, identify what type of form control you would use to simplify data entry for the customers.

2. HTML forms are the most important types of Web pages. Do you agree or disagree with this statement and why?

SKILL DRILL

Skill Drills reinforce project skills. Each skill reinforced is the same, or nearly the same, as a skill presented in the lessons. Detailed instructions are provided in a step-by-step format. Work through these exercises in order.

1. Create a Web Form to Sign in for an Online Exam

In this Skill Drill, you create a basic form that serves as the opening page for an online examination for the Web Design 301 course. (In Challenge Exercise 1, you add to this series of examination pages.)

1. Create a new site definition called `Casual University` from the Chapter_01>CasualU folder.
2. Open signin.html.
3. Create a form named `signIn`; the Method is POST, the Action is `q1.html`, and the Enctype is set to application/x-www-form-urlencoded.
4. In the form area, insert a `3`-row, `2`-column table with the width set to `60` percent, border thickness set to `0`, cell padding set to `3`, cell spacing set to `0`, and headers set to none. Select the top four cells and set the cell width to `50%`. Select the top and middle left cells and set the cell alignment to right.

5. In the top-left cell, type `Student Name`, and in the cell below, type `Student ID`. In the top-right cell, insert a text field named `studentName` that is `35` characters wide. When prompted for a label, click Cancel. In the middle-right cell, insert a password field named `studentID` that is `35` characters wide. When prompted for a label, click Cancel.
6. Merge the bottom two cells. Set the horizontal alignment of the merged cell to center. In the merged cell, insert a Submit button with the name `signIn` with a value of `Sign Me In`. When prompted for a label, click Cancel.
7. Right/Control-click the studentName text field, and select Edit Tag <input>; in the Style Sheet/Accessibility category, ensure that the ID is `studentName` (add it if it is not). Using the same procedures, ensure the studentID text field has the ID of `studentID`.
8. Select Student Name in the left cell, Right/Control-click, and use the Quick Tag Editor to insert the `<label>` tag, with the `for` attribute set to `studentName`. Using the same procedures, enclose the Student ID in the `<label>` tag, setting the `for` attribute to `studentID`.
9. Close signin.html, saving your changes.

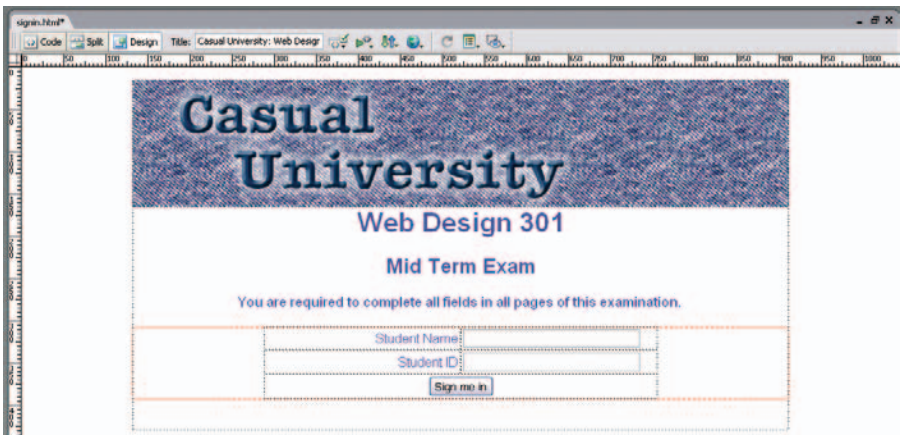

FIGURE 1.60

2. Create a Contact Form

A contact form is one of the most common means of contacting a company. It provides the employees some degree of protection against spam by hiding their e-mail from direct access. A contact form can accommodate changing staff when a contact person changes position, department, or employers. A contact form can also help categorize the message based on the types of questions the customers/visitors have and the department they want to contact, making it easier for the company to respond.

In this exercise, you create a contact form that includes the basic Name and E-mail fields as well as lists of departments and types of questions the visitor might have.

1. Open contactus.html from the Forms site.

2. In the empty paragraph below the Contact Us heading, insert a form with the name `contactus`, Method set to POST, Action set to `thankyou.html`, and Enctype set to application/x-www-form-urlencoded.

3. In the form, insert a `7`-row, `2`-column table, with the width set to `100` percent, border thickness set to `0`, cell padding set to `2`, cell spacing set to `0`, and headers set to none. Select the left column and set the horizontal alignment to right. Merge the bottom two cells and set the horizontal alignment to center.

4. In the top-left cell, type `Your Name:`, and in the right cell, insert a text field (cancel the creation of a label) with a name of `contactName` and a width of `35`. In the left cell of the second row, type `Your E-mail:`, and in the right cell, insert a text field (cancel the label) with a name of `contactEmail` and a width of `35`. In the left cell of the third row, insert one checkbox (cancel the label) with a name of `ccSender`, a Checked Value of `yes`, and Initial State set to Checked. In the right cell, type `Send a copy of the message to you.`

5. In the left cell of the fourth row, type `Department:`. In the right cell, create a menu (no label) named `department` with the following five items: `Select a department` (assign a value of `noDept`), `Sales`, `Service`, `Human Resources`, and `Web master` (no values for the last four items). Set Select a department as the Initially selected item.

6. In the left cell of the fifth row, type `Reason for Contacting Us:`. In the right cell, create a menu (no label) named `reason`. Using the chart below, create a list of items. When finished, set Choose a reason as the Initially Selected item.

Item Label	*Value*	*Selected*
Choose a reason	noReason	selected
Order not arrived	notArrived	
Order received broken	broken	
Other	otherSales	
Warranty enquiry	warranty	
Schedule a repair	schedRepair	
Order a part	partOrder	
Other	otherService	
Privacy and other policies	policies	
I would like a job	employment	
Other	otherHR	
Broken Web link	brokenLink	
Other	otherWeb	

The fact that there are multiple instances of an option labeled Other requires that they be distinguished using different values.

7. Click to select the Reason menu and switch to the Code view. Select from Order not arrived to otherSales (include both the opening and closing **<option>** tags when selecting). Right/Control-click in the selected region, select Insert tag, and insert the **<optgroup>** tag with a label of **Sales Department**. Select from Warranty enquiry to otherService, and insert the **<optgroup>** tag with the label **Service Department**. Select from Privacy to otherHR, and insert the **<optgroup>** tag with the label **Human Resources**. Select the last two options, and insert the **<optgroup>** tag with the label **Web Site**. Switch to the Design view when finished.

8. In the sixth left cell, type **Comments:**. In the right cell, insert a multi-line text field (no label) with a name of **comments**, a Char Width of **35**, and a Num Lines of **10**.

9. Apply the **<label>** tag to all of the labels, assigning the **for** attribute the same value as the **id** attribute of the related form control. Right/Control-click a form control, select Edit tag, and check for an ID in the Style Sheet/Accessibility category of the Tag Editor dialog box. If an ID does not exist, create one that is identical to the control name (found in the General category of the Tag Editor dialog box). Select the form control's label text, and wrap it with the **<label>** tag, assigning it the **for** attribute identical to the ID attribute of the form control. Repeat these procedures for all six form controls.

10. In the bottom merged cell, insert an image field using the Forms>images>contactusform.gif graphic. Type **submit** in the ImageField field and **Send Your Message** in the Alt field.

11. Preview the page in your browser and click the Image Field button.

    A thank you message is very respectful of your customers. It may contain a static message, like this one, or, through the magic of the form-processing application, may contain a copy of the message they sent via the form.

12. Close your browser, return to Dreamweaver, and close contactus.html.

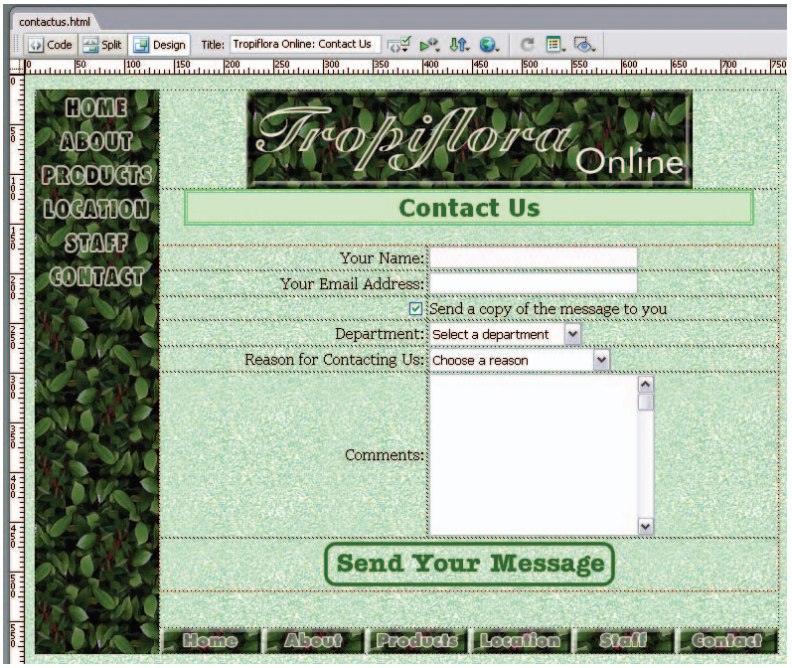

FIGURE 1.61

TO EXTEND YOUR KNOWLEDGE . . .

ONE LIST AFFECTS ANOTHER

This form uses optgroups to help the user select a reason that matches the department selected from the other list. However, this is a perfect example of a form in which JavaScript could be used, so instead of optgroups, the selection of a department from the first list would hide all reasons except those that apply to the selected department.

3. Create a Marketing-Survey Form

In this exercise, you create a market-survey form using Radio button groups to provide the options while restricting the user to selecting just one from each group.

1. Create a new HTML page, title it `Market Survey`, and save it (but do not close it) as `survey.html` in the Forms site.
2. Type `Market Survey`, format it as Heading 1, and press Enter/Return.
3. Insert a form with the name `survey`, the Method set to POST, Action set to `market-survey.html`, and Enctype set to application/x-www-form-urlencoded.
4. Type `Select your age group` and press Enter/Return. Create a Radio Group called `age`. Create four Radio buttons labeled `Under 25`, `25 to 34`, `35 to 44`, and `45 and older`. Make up your own values for the four Radio buttons, keeping them short and without spaces. If necessary, use the Plus button to add new rows and the up and down arrows to sort the entries from youngest at the top to oldest at the bottom. Lay out the buttons using a table.
5. Click below the Radio buttons table, type `Select your family income`, and press Enter/Return. Create a Radio button group called `income` using `$30,000 or Less`, `Over $30,000 to $45,000`, `Over $45,000 to $60,000`, and `Over $60,000`, and create your own values to match the labels. Lay out the buttons using a table.
6. Click below the Radio buttons table, type `Select your dwelling type`, and press Enter/Return. Create a Radio button group called `dwelling` using `Apartment`, `Row House-Rented`, `Row House-Owned`, `Condominium`, `Duplex`, and `House` as the labels, and create your own values. Lay out the buttons using a table.
7. Click below the Radio buttons table, type `Select your current profession`, and press Enter/Return. Create a Radio button group called `profession` using `Retail`, `Government`, `Health Services`, `Business Owner`, and `Other`, and create your own values. Lay out the buttons using a table.
8. In the last cell to the right of Other, click and press Enter/Return. Insert a text field using `If you selected Other, please specify:` as the Label text. Name the text field `specifyOther` and set its width to `25`.
9. Insert a Submit button, and set it so that it displays `Thank You!` on the button.
10. Close survey.html, saving your changes.

4. Create a Book Wish-List Form

For the purpose of this exercise, pretend you have created an application in which you, your family, and your friends may add entries to a Book Wish-List database. If anyone is trying to think of a perfect birthday gift, all they need to do is consult the database. In this exercise, you create the front-end form that will submit the book details to the database. As part of the form, you add the URL to the Web site of the book (if there is one) and insert a little JavaScript function with which you may test the book Web-site URL.

1. Create a new HTML page, title it `Book Wish List`, and save it as `wishlist.html` in the Forms site.
2. Type `Book Wish List` and format it as Heading 1. Press Enter/Return, type `Add a new entry`, and format it as Heading 2. Press Enter/Return and type `All fields marked with an asterisk must be filled.` Select this paragraph and apply bold. Press Enter/Return.
3. Create a form with the name `wishlist`, Action set to `wishlist.html`, Method set to POST, and Enctype set to application/x-www-form-urlencoded.
4. Create a `7`-row, `2`-column table with the width set to `60` percent, the border thickness set to `0`, cell padding set to `2`, cell spacing set to `0`, and no headers. Select the left column of cells, and set the horizontal alignment to right. Merge the two cells in the bottom row, and set the horizontal alignment to center.
5. In the left column, type `Title*`, `Author(s)*`, `ISBN`, `Publisher*`, `Publication Date*`, and `Book Web Site`. In the right columns, insert text fields for the `title`, `author`, `ISBN`, and, `publisher` with appropriate names and widths but do not create labels for them. Create a text field for the book Web site and name it `bookURL`. (The name is important as it is used in a JavaScript function added in Step 8.)
6. To enable the user to insert the publication date, create two menus. The first menu is a list of the months of the year, so label and name it appropriately. Above `January`, insert `Month`, assign it an appropriate value, and then add the rest of the month names (with or without values, your choice). The second menu is a list of years starting with `Year`, then listing `2004` backwards to `2000`. Assign an appropriate label and name for this list and also an appropriate value for Year, the top item in the list.
7. In the merged cell, insert a Submit button that displays `Add To Wish List`.
8. Open testurl.html, press Control/Command-A to select all of the content, copy it to the clipboard, and close testurl.html. Click to the right of the book Web-site text field, and paste the copied code.
9. Preview the page in your browser.
10. Complete the form as follows: `Speed Up Your Site` is the title, `Andrew B. King` is the author, `0-7357-1324-3` is the ISBN, `New Riders` is the publisher, `January 2003` is the publication date, and `http://www.websiteoptimization.com` is the book's Web site. Click the Test URL link and observe that the book's Web site opens in a pop-up window.
11. Close the pop-up window, close your browser, return to Dreamweaver, and close wishlist.html.

CHALLENGE

Challenge exercises expand on, or are somewhat related to, skills presented in the lessons. Each exercise provides a brief narrative introduction, followed by instructions in a numbered-step format that are not as detailed as those in the Skill Drill section. You can work through one or more exercises in any order.

1. Create a Multiple-Choice Exam

In this challenge, you create three forms containing questions from a multiple-choice exam. The action URLs send the student to the next question. Hidden fields contain the answers to previous questions. Although using hidden fields this way may not be very secure, it is possible to ***obfuscate*** (make obscure or unclear) the answer using a ***hash*** (a process of encrypting text whereby the result cannot be reversed to reveal the original text but two encryptions of the same text produce the same result, enabling comparison) of the answer so that the student cannot simply change the answer in the hidden field from A to B. This exam process, and the generation of the hash (which is an available function in PHP and other programming languages), would normally be done with a form-processing application. However, this exercise will provide you with an understanding as to the type of process one might apply to a form-processing application.

1. Open q1.html from the Casual University site, and click in the empty paragraph below the instructions.
2. Create a new form called `q1` with the Action set to `q2.html`, the Method set to POST, and the Enctype set to application/x-www-form-urlencoded.
3. In the form, type `1. Which text format displays using the largest text size?` and press Enter/Return. Create a Radio button group with the name `q1`. Type `Heading 6`, `Heading 2`, `Paragraph`, and `Heading 1` as the labels, and `a`, `b`, `c`, and `d` as the values. Lay out the Radio button group using line breaks.
4. Click in the empty paragraph after the last answer and insert a hidden field with `studentID` as the name and `25D55AD283AA400AF464C76D713C07AD` as the value.

   This 32-character string is a hash of 12345678.
5. Create a Submit button with `next` as the name and `Next Question` as the value.
6. Close, saving the changes.
7. Open q2.html. Use the same procedures as you did in Steps 2 through 6 with the following changes: set the form name as `q2` and the form's action to `q3.html`. The question is `2. Absolute links are necessary when linking:` and the answers are `a page to a named anchor`, `to a target window`, `to a page in another Web site`, and `alternate text`. Insert the same studentID hidden field (you may wish to open q1.html and copy and paste the hidden button to this page) plus another hidden field called `answer1` with a value of `8277E0910D750195B448797616E091AD`. When finished, close q2.html, saving your changes.
8. Open q3.html. Use the same procedures as in Step 7 with the following changes: set the form name as `q3` and the form's action to `results.html`. The question is `3. GIF and JPEG image formats:` and the answers are `are bitmapped graphics`, `are limited to 256 colors`, `can be animated`, and `are the storage formats of digital cameras`. Insert the same `studentID` and `answer1` hidden fields plus another hidden field

called `answer2` with a value of `4A8A08F09D37B73795649038408B5F33`. Set the button to display `See Results!`. When finished, close q3.html, saving your changes.

9. Open results.html. In the empty cell below Results, insert a Definition List from the Text Insert bar. Type `1. Which text format displays using the largest text size?`, press Enter/Return, type `Heading 1`, and press Enter/Return. Type `2. Absolute links are necessary when linking:`, press Enter/Return, type `to a page in another Web site`, and press Enter/Return. Type `3. GIF and JPEG image formats:`, press Enter/Return, type `are bitmapped graphics`, and press Enter/Return twice.
10. Type `Congratulations! Perfect!` and press Enter/Return once.
11. Create a form named `results` with no Action or Enctype settings. Insert the same three hidden fields as you did previously: `studentID`, `answer1`, and `answer2`. Create a fourth hidden field called `answer3` with a value of `0CC175B9C0F1B6A831C399E269772661`. Close, saving the changes.

12 Open signin.html and preview it in your browser. Sign in and answer the questions in the forms. When finished, close your browser, return to Dreamweaver, and close signin.html.

TO EXTEND YOUR KNOWLEDGE . . .

A HASH CALCULATOR

If you search for hash calculator using your favorite search engine, you will find a number of downloadable, free calculators that enable you to calculate different forms of hashes. You may also go to http://bfl.rctek.com/tools/?tool=hasher for an online version. When you compare the MD5 hash for the letters a, d, and c (individually), you see that these values are equal to the values of the answer1, answer2, and answer3 hidden fields.

2. Create a Two-Form Page

When working with forms, one common programming technique is that when two or three forms are related to each other, all two or three forms are created within the same HTML page. The form data sent by the first form is used by the programming code to determine that the first form has been completed and the second form is to be displayed. For example, the first form could ask for your name. When the form is sent back to itself, the programming code sees that your name has been completed and sends you the second form (with your name stored in a hidden field). The second form asks you for your gender. When the form is sent back to itself, the programming code takes your name from the hidden field and your gender from the most recent form data and creates a suitable message for you.

In this exercise, you create both forms on the same page and insert some ***pseudo-code*** (programming-like code written in understandable English instructions to describe the process that will be used) that, on a Web server, might be used to populate the employee-details form with the information about the employees.

1. Open edit-student.html from the Casual University site, and click in the empty cell below the top graphic.
2. Type `Select a Student` and format it as Heading 1. Press Enter/Return.
3. Insert a form with a name of `selectStudent`, the Action set to `edit-students.html`, the Method to GET (for the time being so we can analyze the query string), and Enctype to application/x-www-form-urlencoded.
4. Insert a menu with a label `Select a student` and `studentID` as its name with the following five student names and values (student IDs): `Carole Brown` (`1234`), `Bruce Carter` (`2345`), `Jane Higgins` (`3456`), `Tommy Kettle` (`4567`), and, finally, your own name and a four-digit number.
5. Click to the right of the menu control and press Enter/Return. Insert a Submit button with the name `submit` and a label of `Edit Student Record`.
6. In the empty paragraph below this form, type `Edit Student Record` and format it as Heading 1. Press Enter/Return and create another form named `editStudent`, set Action to `edit-student.html`, the Method to GET, and Enctype to application/x-www-form-urlencoded.
7. In the form, create a `9`-row, `2`-column layout table with a width of `60` percent, a border width set to `0`, cell padding set to `2`, and cell spacing set to `0`. Right-align the left column. In the left column, type the following texts: `Student ID`, `First Name`, `Last Name`, `Street Address`, `City/Town`, `State`, `Home Phone`, and `Birth Date`. In the right column, insert text fields for the first seven rows and then make the text in the left cells the explicit labels for the text fields. Create three text fields for Birth Date with the labels `Day`, `Month`, and `Year`. Merge the bottom row, center align the cell, and insert a Select button with the name `submit` and the label `Save Changes`.
8. Insert your name and particulars as the initial values for all of the fields.
9. In the empty paragraph below the second form, type `Student Record Updated` and format it as Heading 1.
10. Switch to the Code view and scroll to approximately line 17, where `<h1>Select a Student<h1>` appears. Open edit-student.php.txt, select lines 2 through 6, copy the text, switch to edit-student.html, and paste the text to the left of the `<h1>` tag around line 17. Switch to edit-student.php.txt and select lines 10 through 17, copy them, switch back to edit-student.html, and paste them to the left of the `<h1>` tags near line 37. Switch to edit-student.php.txt, copy lines 21 through 26, switch back to edit-student.html, and paste them to the left of the last `<h1>` tag near line 92.

    Read the three inserted blocks of pseudo-code. The first block says that if there is no query string, the first form is displayed. The second block says that if the query string identifies that the Edit Student Record button is clicked and contains the studentID, the database will be queried using that studentID and the second form is populated with the student's details and displayed. The third block says that if the Save Changes button was clicked, the database is updated with the new data and the third part of the page is displayed.

11. Preview the page in your browser. Select your name from the top form, click the Edit Student Record button, and examine the query string for submit=Edit+Student+Record and your studentID. Click the second Submit button and examine the query string for submit=Save+Changes.

    In this situation, it is fine to have three Heading-1 blocks because only one displays at a time.

12. Close your browser, return to Dreamweaver, and close edit-student.html and edit-student.php.txt.

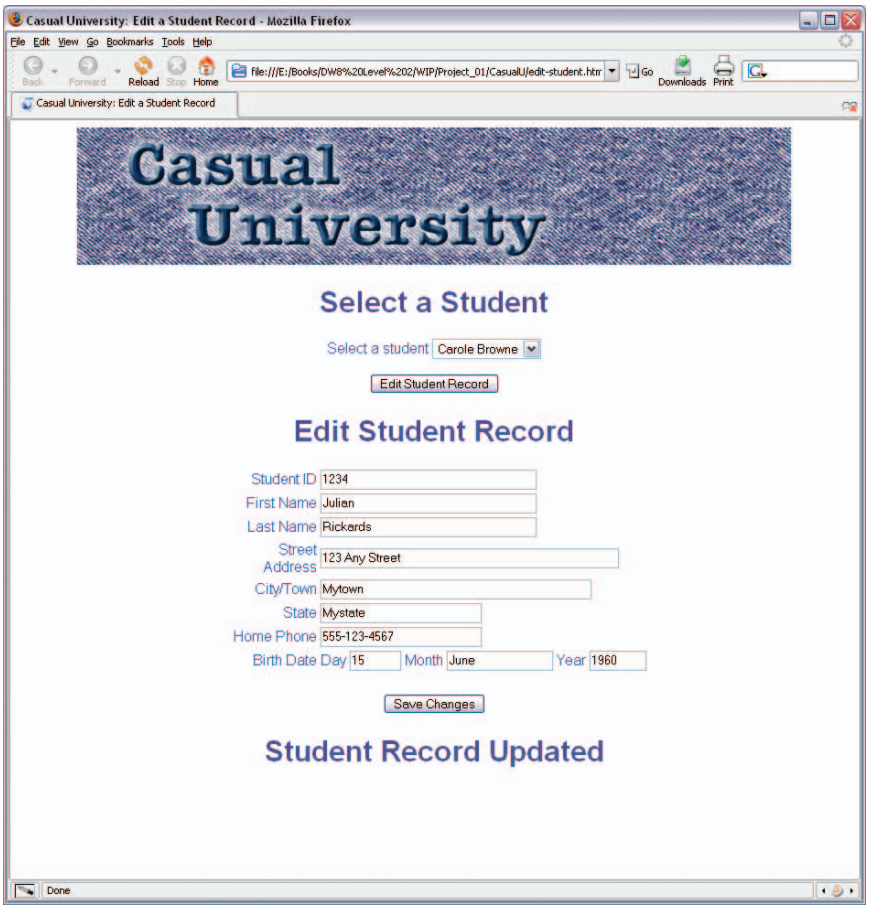

FIGURE 1.62

TO EXTEND YOUR KNOWLEDGE . . .

CHALLENGES OF CREATING FORMS

Forms can be difficult to create. The first issue deals with the language of the labels and fields. For example, if you are 25 years old and a form offers "20 to 25" and "25 to 30" age options, which one do you select? This is not an area with which we can help other than to recommend that you perform user testing to help identify some of these issues.

The other issue with the creation of forms is the effort one must expend to set up the form controls and their layout. This is especially difficult for people who do not know Web-page design, and they will often turn to online services to assist them. Unfortunately, some of these form creation services, while they may create forms according to your specification, depend on a lot of JavaScript to the point where a JavaScript-free or -disabled browser would not be able to use the forms. Furthermore, the accessibility of these forms is suspect: we are aware of one user who was not able to complete a form (survey) generated by one of the more popular form-generation services because her screen-reader software would not work with the form.

Form creation may be challenging, but to ensure that the forms meet your criteria, you should invest the time and effort to create them properly. Furthermore, many forms, such as contact forms and registration forms, rarely require changes: once you have created these forms, you often do not have to revisit them for a long time.

3. Using Image Buttons to Submit Orders

The Image button is useful, not only for decorating the text of a Submit button but also, on a shopping site, for submitting an item to the ***shopping cart*** (a term used to describe the shopping process used on an online store) for purchase. In this exercise, you complete the form that students may use to purchase books for their courses. In place of plain buttons, you use image buttons that are images of the book covers.

1. Open order-books.html from the Casual University site.
2. Click in the cell to the left of the description of the Dreamweaver book. Click the Image Field button from the Forms Insert bar, and insert Dreamweaver_Cover.JPG from the images folder. Type `Order this Dreamweaver book` in the alternate-text field. Copy the ISBN number from the right cell, and paste it into the name field of the Image Field button.
3. Repeat Step 2 and insert Image Field buttons for each of the books. Change the alternate text to reflect the book, and change the name of the field to reflect the ISBN number.

   The ISBN number is a unique identifier of each book and is used by the database system to manage the shopping cart as well as other business functions of the online store.
4. Preview the page in your browser, click any of the book covers, and examine the query string.

   Image Field buttons appear twice in a query string because they indicate not only which image was clicked but also the x and y position of the mouse pointer where it was clicked. In some other applications, the position of the mouse pointer is useful, but for this type of application, all that is needed is the identity (ISBN) of the selected book.
5. Close your browser, return to Dreamweaver, and close order-book.html.

4. Hide Optional Fields Using JavaScript

You have read often in this chapter that JavaScript is commonly used to enhance the usability of forms. In this exercise, you create a form that has both required and optional fields. The JavaScript function in this page, when clicked, hides the optional fields and, when clicked again, reveals the optional fields. You do not need to insert or type any of the JavaScript; it is configured to act on certain classes and IDs. When you create the optional fields in this form to enable JavaScript to work on them, you must assign them the class name optional. The JavaScript code looks for all instances of the class optional and hides or reveals them depending on how many times the link is clicked. The link also is dynamically created using JavaScript. In Dreamweaver, the link is not visible because Dreamweaver does not run the JavaScript.

1. Open contactus.html from the Casual University site.
2. Click to the right of Contact Us and press Enter/Return.
3. Press Control/Command-B to apply bold, type `Fields marked with the asterisk are required.`, press Control/Command-B to disable bold, and press Enter/Return.
4. Right/Control-click the empty paragraph, and select Edit Tag <p> from the contextual menu. From the Style Sheet/Accessibility category of the Tag Editor dialog box, type `hidedisplaylink` in the ID field (the spelling and letter case is important or the JavaScript will not be able to act on it).

? If You Have Problems

If you have problems Right/Control-clicking the empty paragraph, type some text and then Right/Control-click and complete the rest of Step 4. Delete the text but not the paragraph tag when you are finished with Step 4.

5. Create a form with the name `contactus`, the action set to `thankyou.html`, the method set to POST, and Enctype set to application/x-www-form-urlencoded.
6. In the form, create a `6`-row, `2`-column layout table with a width of `60` percent, a border width set to `0`, cell padding set to `2`, and cell spacing set to `0`. Right-align the left column and left-align the right column. In the left column cells, insert the following texts: `Full Name`, `Student ID`, `E-mail address`, `Subject*`, and `Comments*`. In the right columns, create appropriate text fields for each of the texts in the cells to the left, except the Comments field. The Comments field should be a text-area field, `45` characters wide and `12` lines high. Set the texts in the left cells to be the labels of the text and textarea fields.
7. Merge the bottom two cells, center align the merged cell, and insert a Submit button with the label `Send us your comments`.
8. Click in Full Name, and, using the Tag selector, click the **<tr>** tag. Right/Control-click the **<tr>** tag, select Quick Tag Editor, add `class="optional"` to the tag, and press Enter/Return (do not forget to add a space before typing class). Repeat the process for the **<tr>** tag of the Student ID and E-mail address rows.

9. Preview this page in your browser.

   Notice the link that was not visible in Dreamweaver. The JavaScript function has been programmed to create this link and text dynamically.

10. Type information in all of the fields.

11. Click the link and notice that the optional fields are hidden (by hiding the table rows in which they exist) and the link-text changes, enabling you to reveal the optional fields.

    Notice that the contents of the fields are not erased when the fields are hidden and revealed again. The hiding processing does not erase the table rows and its contents; it hides them.

12. Close your browser, return to Dreamweaver, and close contactus.html.

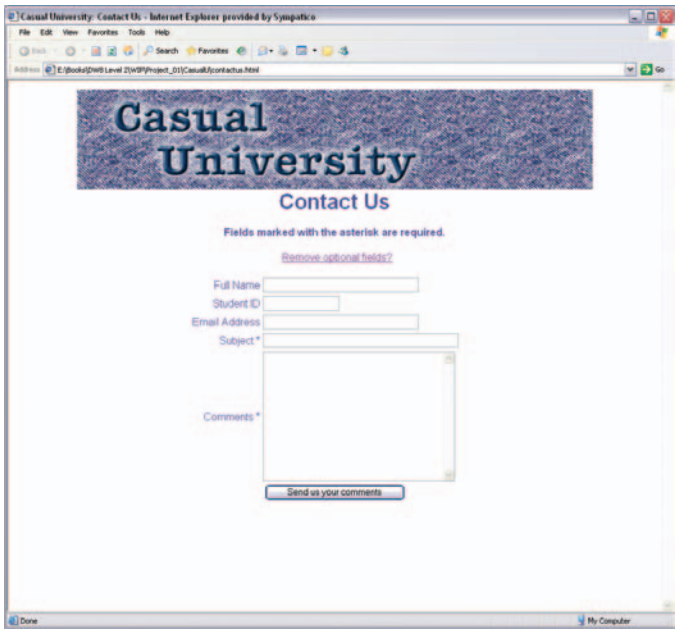

FIGURE 1.63

PORTFOLIO BUILDER

Create Forms to Process a Purchase

Online stores require a large amount of information to enable customers to complete their purchases. In this assignment, you are going to create a series of forms that collect and process customer and purchase information for an online bookstore.

Before you begin, write down the author and title of three books you own, and use these as the selected products from your online bookstore. Assume that the books you have chosen are stored in a shopping cart database. Consider the following options when you create the forms:

1. The customer may be a new or returning customer. Given that you need information about the customer but do not want to ask returning customers to enter their personal information each time they visit your site, create the form or forms necessary to gather information from a new customer. Provide an option to allow returning customers to identify themselves without having to reenter this information.
2. A purchase-confirmation page allows customers to view their purchase selections and, if necessary, reconsider their selections. Create a form that displays their shopping cart items and allows them to make some changes. Consider the type of changes you would allow customers to make to their orders and provide the appropriate options in the form.
3. Create a form that displays the customer's shipping information and provides shipping options.
4. Create a payment form allowing users to select how they will pay for their products. Do you only accept credit cards, or would you also allow other payment methods? What type of information do you need from the customers, and what information might customers need from the store for their selected payment option?
5. Create a page that displays their final invoice. This page is not a form but gathers information from the previous forms. Do not display sensitive information, such as credit card information.