

Objectives

Session 6.1

- Explore how Web forms interact with Web servers
- Create form elements
- Create field sets and legends
- Create input boxes and form labels

Session 6.2

- Creation option buttons
- Create selection lists
- Create check boxes
- Create text area boxes
- Apply styles to Web forms

Session 6.3

- Work with form buttons
- Explore image elements and hidden fields
- Work with form actions and methods

Working with Web Forms

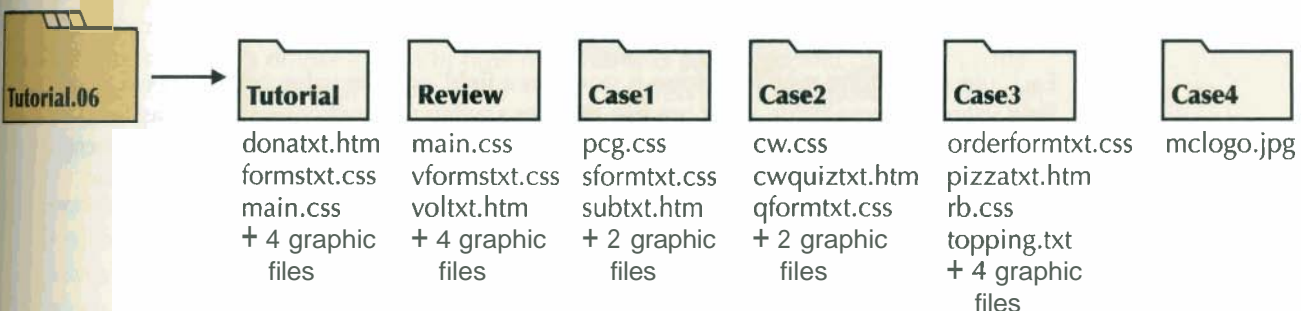
Creating a Donation Form

Case | The Lighthouse

Terry Ives is the director of The Lighthouse, a community center in St. Peters, Missouri. The Lighthouse provides social services, focusing on drug addiction counseling, job placement, child care, and tutoring disadvantaged youths. The Lighthouse's mission is broad and challenging, and as a nonprofit organization, money is always tight.

You've been volunteering at the center for several months, helping to upgrade the Web site's design and adding new features that will make the site more useful for clients, volunteers, and donors. Terry would like your help with one important feature: creating a page for online donations. She knows that many social service organizations receive a good percentage of their donations online; therefore, she's been working with an Internet service provider (ISP) to find out how to facilitate secure online donations. She has learned that the donations page needs a Web form that can be used to transfer payment data to the ISP's Web server for processing. Terry has asked you to create a Web form that will supply the server with the needed financial data.

Starting Data Files



Session 6.1

Introducing Web Forms

You meet with Terry to discuss the new donations page for The Lighthouse's Web site. She sketches out the appearance of a form shown in Figure 6-1 that she would like to display on the center's Web site.

Terry's proposed donations form

The form is divided into three topical areas. The first requests contact information from the donor, including the donor's name, phone number, and mailing address. In the second part of the form, the donor specifies the amount of the donation and provides credit card information. The final part of the form is reserved for any comments the donor has and offers a check box where donors can indicate an interest in volunteering at the center.

Parts of a Web Form

Each piece of information for a form is stored in a **field**, and the value itself is known as the **field value**. In some fields, users are free to enter anything they choose, while other fields are limited to a set of possible values. Users enter or select a field value using **control elements**, which are buttons, boxes, lists, and so on, that provide a way of associating a field value with a particular field. HTML supports the following control elements:

- **input boxes** for text and numerical entries

- **option buttons**, also called radio buttons, for selecting a single option from a predefined list
- **selection lists** for long lists of options, usually appearing in a drop-down list box
- **check boxes** for specifying yes or no
- **text areas** for extended entries that can include several lines of text

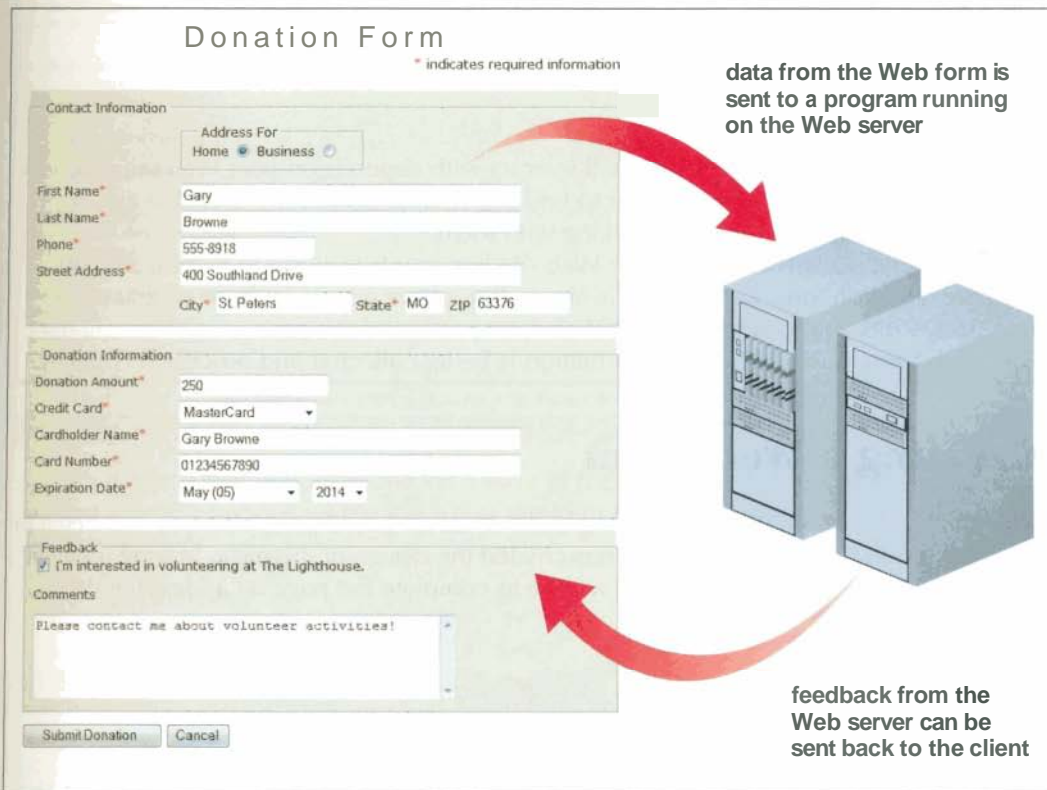
Terry's donation form includes several examples of these different control elements, each one associated with a particular field. As you'll see later, users will enter their first and last names in the `firstName` and `lastName` fields through the use of an input box control element. They'll indicate their credit card through the use of a selection list.

Forms and Server-Based Programs

Before you start work on Terry's Web form, you should understand how forms are processed on the Web. As shown in Figure 6-2, the Web form is used to collect information, but the data itself is stored and analyzed using a program running on a Web server.

The interaction between the Web form and the Web server

Figure 6-2



The pairing of server-based programs and Web forms early in the development of HTML represented a dramatic shift in how the Web was perceived and used. By giving users access to programs that react to user input, the Web became a more dynamic environment where companies and users could interact. Server-based programs have made many things possible, including:

- online databases containing customer information
- online catalogs for ordering and purchasing merchandise
- dynamic Web sites with content that is constantly modified and updated
- message boards for hosting online discussion forums

Because these programs run on Web servers, rather than locally, you might not have permission to create or edit them. Instead, you'll receive information about how to interact with the programs on the Web server. This usually includes a list of fields that are required by the program and a description of the type of values expected in those fields. The Web form code needs to work in conjunction with the requirements of the server-based program.

There are several reasons to restrict direct access to these programs. The primary reason is that when you run a server-based program, you are interacting directly with the server environment. Mindful of the security risks that computer hackers present and the drain on system resources caused by large numbers of programs running simultaneously, system administrators are understandably careful to maintain strict control over their servers and systems. Otherwise, people could use malicious code to inject programming into the server and possibly change the prices of items or degrade the performance of the server.

Server-based programs are written in a variety of languages. The earliest and most common of these languages are called **Common Gateway Interface (CGI) scripts**, written in a language called **Perl**. Other popular languages widely used today for writing server-based programs include:

- ASP
- ColdFusion
- C/C++
- PHP
- VBScript

Which language your Web form will interact with depends on your Web server. Check with your ISP or system administrator to find out what programs are available and what rights and privileges you have in working with them.

The ISP that hosts The Lighthouse's Web site has scripts in place to receive the data from the donation form and process it. You will not have access to these programs, so Terry just wants you to work with the Web form portion of this process. Others will test your Web form to verify that the information is being collected and processed correctly.

Creating a Web Form

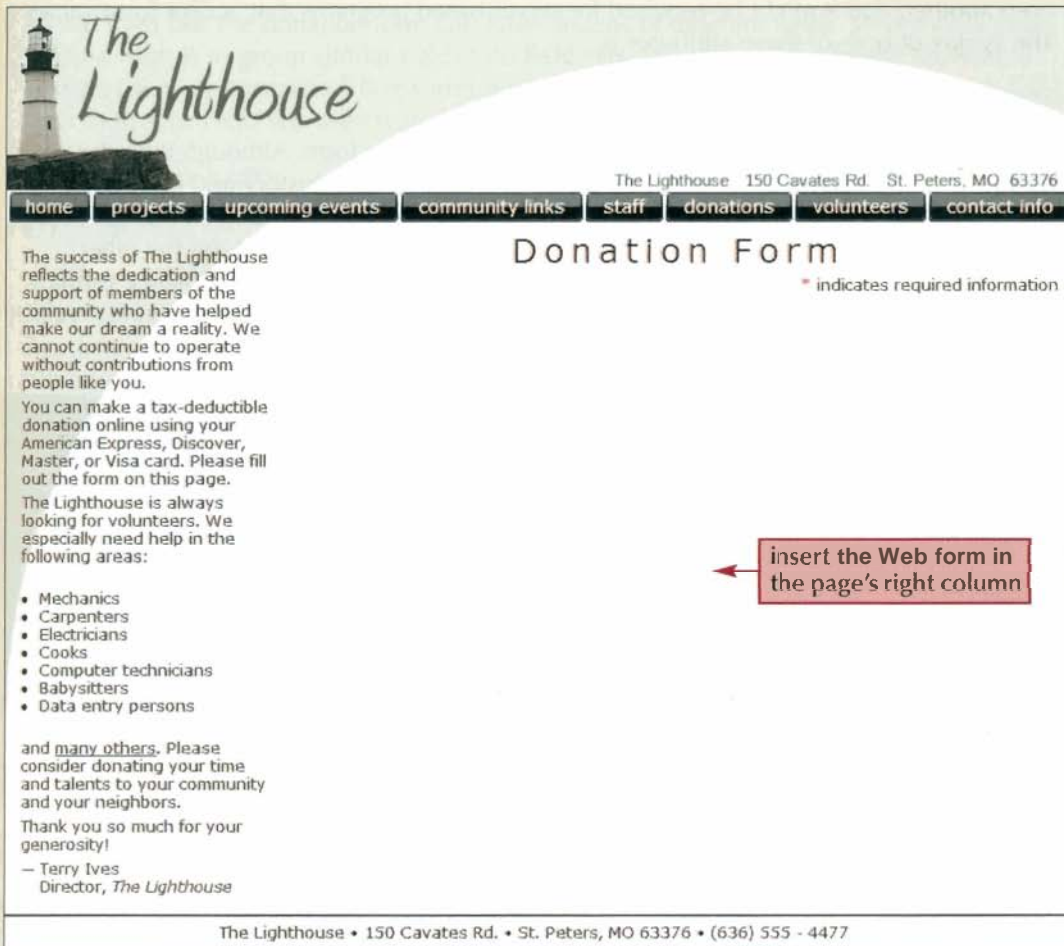
Now that you're familiar with the background of server-based programs, you can begin to work on Terry's donation form. Terry has created the design of the page, leaving the right column empty for the form. Your job will be to complete the page by adding the Web form. Open Terry's document now.

To view Terry's document:

- ▶ 1. Start your text editor, and then open the **donatxt.htm** file located in the `tutorial.06\tutorial` folder included with your Data Files. Enter ***your name*** and ***the date*** in the comment section of the file. Save the file as **donations.htm** in the same folder.
- ▶ 2. Review the file to become familiar with its contents and structure, and then open the file in your Web browser. Figure 6-3 shows the current appearance of the page.

The initial donations page

Figure 6-3



Terry suggests that you insert the Web form in the page's right column. She has already inserted a heading above where she wants the form to appear. Forms are created using the form element

```
<form attributes>
  elements
</form>
```

where *attributes* are the attributes that name the form and control how it is processed, and *elements* are the elements placed within the form. Forms typically contain many of the control elements that were discussed earlier, but can also contain page elements such as tables, paragraphs, inline images, and headings. The form element can be placed anywhere within the HTML file and a single page can contain multiple forms.

Form attributes tell the browser the location of the server-based program to be used on the form's data, how that data is to be transferred to the script, and so forth. These attributes are not needed when first designing the form, and it's actually useful to omit them at first. This prevents you from accidentally running the program on an unfinished form, causing the Web server to process incomplete information. After you've finalized the form's appearance, you can add the attributes required by the server program. You'll have a chance to do this in the last session of this tutorial.

Two attributes identify the form: the `id` attribute and the `name` attribute. Naming a form is useful for pages that contain multiple forms so you can differentiate one form from another, and it might be required for server-based programs that accept form values. The syntax of both of these attributes is

```
<form name="name" id="id"> . . . </form>
```

where `name` is the name of the form, and `id` is the id of the form. Although these two attributes might appear to do the same thing, each has its own history and role. The `name` attribute represents the older standard for form identification, and so is often required for older browsers and older server programs. The `id` attribute, on the other hand, represents the current standard under HTML and XHTML for identifying a form. For maximum compatibility with older and newer browsers and CGI scripts, you should include both attributes, setting them to the same value.

Reference Window | Inserting a Web Form

- To insert a Web form, add the elements

```
<form attributes>
  elements
</form>
```

to the Web page, where **attributes** are the attributes that name the form and control how it is processed, and **elements** are the elements placed within the form.

- To identify the form, add the attributes

```
id="id" name="name"
```

to the opening `<form>` tag, where **id** is the form id and **name** is the form name. You will often set these attributes to the same value.

You are ready to add a form element named `donationForm` to Terry's Web page.

To insert the form element:

- Return to the `donations.htm` file in your text editor and scroll down the file to the `rightColumn` div container.
- Insert the following form element within the `rightColumn` div container, as shown in Figure 6-4.

```
<form name="donationForm" id="donationForm">
</form>
```

Figure 6-4

Inserting a form element

```
div id="rightcolumn">
  <h1>Donation Form</h1>
  <p><span>*</span> indicates required information</p>
  <form name="donationForm" id="donationForm">
  </form>
</div>
```

With the form element added to the donations page, you can start populating it with control elements and other form features. You'll start by adding field sets.

Creating a Field Set

A Web form like the donation form can have dozens of different fields. One way of organizing a form is to group similar fields into **field sets**. When rendered by the browser, a field set usually appears as a box surrounding the fields, separating those fields from other field sets. Field sets are created using the `fieldset` element, which has the syntax

```
<fieldset id="id">
  controls
</fieldset>
```

where *id* identifies the field set and *controls* are the control elements associated with fields within the field set. The *id* value is not required, but it is useful in distinguishing one field set from another. Terry wants to organize the donation form into three field sets named `contact`, `donation`, and `feedback`. Add these field sets to her donation form.

Tip

Field sets make it easier for users with aural browsers and screen readers to navigate your Web form.

Creating a Field Set

Reference Window

- To create a field set, add the element


```
<fieldset id="id">
  controls
</fieldset>
```

 to the form, where *id* identifies the field set and *controls* are the control elements associated with fields within the field set.

To insert a field set:

- Return to the `donations.htm` file.
- Within the form element, insert the following three field sets, as shown in Figure 6-5:

```
<fieldset id="contact">
</fieldset>

<fieldset id="donation">
</fieldset>

<fieldset id="feedback">
</fieldset>
```

Inserting field sets

Figure 6-5

```
<form name="donationForm"
  <fieldset id="contact">
  </fieldset>
  <fieldset id="donation">
  </fieldset>
  <fieldset id="feedback">
  </fieldset>
</form>
```

- Save your changes to the file.

Every field set can contain a legend describing its contents. The syntax of the legend element is

```
<legend>text</legend>
```

where text is the text of the legend. The legend element can only contain text and not other page elements. Based on Terry's sketch from Figure 6-1, you'll add the legends Contact Information, Donation Information, and Feedback to the three field sets you created.

To insert legends for the field sets:

- ▶ 1. Return to the **donations.htm** file.
- ▶ 2. Within the first field set, insert the following legend element:


```
<legend>Contact Information</legend>
```
- ▶ 3. In the second field set, insert the following legend element:


```
<legend>Donation Information</legend>
```
- ▶ 4. In the last field set, insert the following legend element:


```
<legend>Feedback</legend>
```

Figure 6-6 highlights the revised text of the HTML file.

Figure 6-6

Creating field set legends

```
<form name="donationForm" id="donationForm">
  <fieldset id="contact">
    <legend>Contact Information</legend>
  </fieldset>
  <fieldset id="donation">
    <legend>Donation Information</legend>
  </fieldset>
  <fieldset id="Feedback">
    <legend>Feedback</legend>
  </fieldset>
</form>
```

- ▶ 5. Now you can view the three field sets in your Web browser. Save your changes to the file, and reload the **donations.htm** file in your Web browser. Figure 6-7 shows the current appearance of the form.

Figure 6-7

Appearance of the field set and legend elements

Donation Form

* indicates required information

legend

field sets

- Contact Information
- Donation Information
- Feedback

Field sets are block-level elements that expand to accommodate their content. Currently, there are no control or other page elements within the three field sets, so the field set boxes are small and narrow. By default, browsers display the legend text in the upper-left corner of the field set box. However, you can use the CSS positioning styles to move the legend position. Terry does not need you to modify the legend, so you'll leave it in its default position. Now that you've created the three field sets, you can begin to populate them with form control elements.



Creating Input Boxes

Most of the control elements in which users either type or select a data value are marked as input elements. The general syntax of this element is

```
<input type="type" name="name" id="id" />
```

where type specifies the type of input control, and the name and id attributes provide the field's name and id, respectively. As with the form element, you should provide both the name and the id attributes, setting them to the same value to ensure compatibility with older browsers. HTML supports 10 different input types, which are described in Figure 6-8. If no type attribute value is specified, the browser will assume a type value of text.

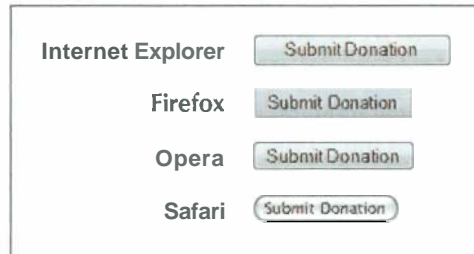
Appearance of control elements **Figure 6-8**

Type Value	Description	General Appearance
button	Displays a button that can be clicked to perform an action from a script	
checkbox	Displays a check box	<input checked="" type="checkbox"/> <input type="checkbox"/>
file	Displays a Browse button to locate and select a file	donations.htm <input type="button" value="Browse..."/>
hidden	Creates a hidden field, not viewable on the form	
image	Displays an inline image that can be clicked to perform an action from a script	
password	Displays an input box that hides text entered by the user	<input type="password" value="....."/>
radio	Displays an option button	<input checked="" type="radio"/> <input type="radio"/>
reset	Displays a button that resets the form when clicked	<input type="button" value="Cancel Donation"/>
submit	Displays a button that submits the form when clicked	<input type="button" value="Submit Donation"/>
text	Displays an input box that displays text entered by the user	<input type="text" value="Terry Ives"/>

The exact appearance of each control element varies among browsers and operating systems. Figure 6-9 highlights the differences among four major browsers in how they render a control button. Because of this variation, you should not rely on the exact appearance of any particular control element when designing your Web form.

Figure 6-9

Control elements under different browsers



When a form is submitted to the server, the server program receives the data in **name/value pairs** in which the name or id of each field is paired with whatever field value is entered into the corresponding control element. The program then processes the data according to each name/value pair. Some server-based programs require a particular field or group of fields. For example, a CGI script whose purpose is to register users might require e-mail addresses entered into a field named e-mail. This means that before specifying a name or id value for a control, you have to learn what the server program expects that data to be named and write your HTML code accordingly. Be aware that case is usually important in specifying field names. A program might not interpret a field named e-mail in the same way as a field named E-MAIL.

The first controls you'll add to the donation form will be input boxes in which donors can enter their first and last names. The syntax to create an input box is:

```
<input type="text" name="name" id="id" />
```

You ask Terry for the ids of the first and last name fields in her donation form. She checks with the ISP hosting The Lighthouse's Web site and tells you that fields containing the donor's first and last name should be given name and id values of firstName and lastName, respectively. You can add these two input boxes to the Contact Information field set. To describe these input boxes for the user, you'll insert the text First Name and Last Name before the input boxes.

Reference Window | Inserting a Text Input Box

- To create a text input box control, use the element `<input type="text" name="name" id="id" />` where the name and id attributes identify the field associated with the input box.

To insert the input boxes:

- Return to the `donations.htm` file in your text editor and scroll down to the Contact Information field set element.
- Within the field set element, add the following text strings and input elements, as shown in Figure 6-10.

```
First Name
<input type="text" id="firstName" name="firstName" />
Last Name
<input type="text" id="lastName" name="lastName" />
```

Adding input box controls

```
<form name="donationForm" id="donationForm">
  <fieldset id="contact">
    <legend>Contact Information</legend>
    First
    <input type="text" id="firstName" name="firstName" />
    Last name
    <input type="text" id="lastName" name="lastName" />
  </fieldset>
```

text type indicates
an input text box

- ▶ 3 Save your changes to the file, and then reload **donations.htm** in your Web browser. Your browser should show two input boxes within the Contact Information field set box. Terry suggests that you test the input boxes.
- ▶ 4. Type **your first name** in the First Name box, press the **Tab** key, and then type **your last name** in the Last Name box. Figure 6-11 shows the input boxes with the sample text.

Input controls with sample data

Contact Information

First Name Terry Last Name Ives

Donation Information

Feedback

HTML treats all form control elements as inline elements, so the input boxes that you created for Terry's form appear within the same line rather than in separate blocks. You can change this by applying the CSS display style to the input box.

Adding Field Labels

In the last set of steps, you entered descriptive text alongside the input boxes to indicate the purpose of the input box to the user. However, nothing in the HTML code explicitly associates that text with the input box. To associate text with a control element, you can use the label element

```
<label for="id">label text</label>
```

where *id* is the value of the *id* attribute for the field's control element, and *label* text is the text of the label. The *for* attribute associates the text of the label with the control element *id*. For example, the following code associates the label text First Name with the first-name control element:

```
<label for="firstName">First Name</label>
<input type="text" id="firstName" />
```

Using the `for` attribute explicitly associates the label with the control element. You can also make this association implicitly by nesting the control element within the label as in the following code:

```
<label>
  First Name
  <input type="text" id="firstName" />
</label>
```

Notice that you do not need to include a `for` attribute when you nest the control element within the label element.

Which approach you take depends on how you want to lay out the form's contents. When you use the `for` attribute, you can place the label text anywhere within the Web page and it will still be associated with the control element. However, by nesting the control element within the label, you can treat both the control element and its label as a single object, which might make it easier to do form layout as you can move both label text and the control element around the page. Depending on the layout of your Web form, you might use both approaches.

Reference Window | Creating a Field Label

- To explicitly associate a text label with a control element, use the label element `<label for="id">label text</label>` where ***id*** is the id of the control element.
- To implicitly associate a text label with a control element, nest the control element within the label as follows

```
<label>
  label text
  control
</label>
```

where ***control*** is the control element. You do not have to include a `for` attribute.

For the `firstName` and `lastName` fields, you'll use the second approach, in which the control elements are nested within their label elements.

To insert the field labels:

- ▶ 1. Return to the `donations.htm` file in your text editor.
- ▶ 2. Enclose the First Name and Last Name text strings within opening and closing `<label>` tags. Indent your code to make it easier to read, as shown in Figure 6-12.

Figure 6-12 Adding field labels

```
<fieldset id="contact">
  <legend>Contact Information</legend>
  <label>
    First Name
    <input type="text" id="firstName" name="firstName" />
  </label>
  <label>
    Last Name
    <input type="text" id="lastName" name="lastName" />
  </label>
</fieldset>
```

- ▶ 3. Save your changes to the file, and reload the page in your Web browser.
- ▶ 4. Test the Labels by clicking each label and verifying that the cursor appears within the corresponding control element.

Working with Form Styles and HTML Attributes

Terry stops by to see your progress on the donation form. She would prefer to have the labels placed in one column and the input boxes put in another column, rather than having both strung together in a single line. Placing labels and control elements in separate columns is a common form layout, one that has often been done with Web tables. However, you've learned that the use of Web tables for page layout is frowned upon. So, instead of a Web table, you'll lay out the form using positioning styles placed in an external style sheet. This has the advantage of making it easier to modify the form layout later on because you will not have to modify the markup code in the HTML file.

To create the form style sheet:

- ▶ 1. Use your text editor to open the **formstxt.css** file from the **tutorial.06\tutorial** folder included with your Data Files. Enter **your name** and **the date** in the comment section of the file, and then save it as **forms.css** in the same folder.
- ▶ 2. Return to the **donations.htm** file in your text editor and add the following link to the **forms.css** style sheet directly above the closing `</head>` tag.

```
click href="forms.css" rel="stylesheet" type="text/css" />
```

You decide to change the display style of the label elements from inline to block so that the labels will appear on a separate line from the input boxes. Because this particular style might not apply to other labels in the donation form or on The Lighthouse's Web site, you'll add a class element named `blockLabel` to the label elements having this design format.

To create the `blockLabel` class:

- ▶ 1. Scroll down the **donations.htm** file and insert the class attribute `class="blockLabel"` in the First Name and Last Name labels, as shown in Figure 6-13.

Adding class names to the field labels

Figure 6-13

```
<label class="blockLabel">
  First Name
  <input type="text" id="firstName" name="firstName" />
</label>
<label class="blockLabel">
  Last Name
  <input type="text" id="lastName" name="lastName" />
</label>
```

- ▶ 2. Save your changes to the file!

Next, you'll create a style for the `blockLabel` class of labels. The style will set the display property of the label to `block`, and set the margins to 12 pixels above and below the label and to 0 pixels to the left and right. You'll also place the label using relative positioning, but you will not define any coordinates so that the label stays in its default position in the page flow. The complete style declaration is:

```
label.blockLabel {display: block; position: relative; margin: 12px 0px}
```

The input elements within each label will be placed using absolute positioning 150 pixels from the left margin of the label. The style declaration is:

```
label.blockLabel input {position: absolute; left: 150px}
```

Add these two styles to the `forms.css` style sheet.

To create styles for the `blockLabel` class:

1. Return to the `forms.css` file in your text editor and add the following styles to the style sheet, as shown in Figure 6-14.

```
label.blockLabel      {display: block; position: relative;
                       margin: 12px 0px}

label.blockLabel input {position: absolute; left: 150px}
```

Figure 6-14

Styles for the `blockLabel` labels and input elements

```
label.blockLabel      {display: block; position: relative; margin: 12px 0px}
label.blockLabel input {position: absolute; left: 150px}
```

2. Save your changes to the style sheet, and reload the donations page in your Web browser. Figure 6-15 shows the new layout of the form fields.

Figure 6-15

Revised layout of the form elements

There are some fields in the donation form that the server-based program will require for the donation to be processed. Terry wants you to mark such required fields with a red asterisk. The `firstName` and `lastName` fields are both required, so you'll mark their labels with an asterisk, adding the red style to the `forms.css` style sheet.

To mark the required fields:

1. Return to the `donations.htm` file in your text editor.
2. At the end of the label text for the `firstName` and `lastName` fields, insert the code

```
<span>*</span>
```

as shown in Figure 6-16.

Marking a required field

Figure 6-16

```

<label class="blockLabel">
  First Name<span>*</span>
  <input type="text" id="firstName" name="firstName" />
</label>

<label class="blockLabel">
  Last Name<span>*</span>
  <input type="text" id="lastName" name="lastName" />
</label>

```

- ▶ 3. Save your changes to the file and then return to the **forms.css** style sheet. Add the following style to the bottom of the sheet to display all span elements from the donation form in a red font:


```
#donationForm span {color: red}
```
- ▶ 4. Save your changes to the style sheet and then reload **donations.htm** in your Web browser. Verify that the labels for the **firstName** and **lastName** variables end with a red asterisk.

Always mark the required fields in your Web form so that users know exactly which fields they must enter and which fields are optional.

Terry wants the same style applied to input boxes for each donor's phone number and street address. Both of these fields are required, so you'll also append an asterisk to the field labels.

To insert additional fields to the donation form:

- ▶ 1. Return to the **donations.htm** file in your text editor.
- ▶ 2. Directly below the label for the **lastName** field, insert the following fields and labels as shown in Figure 6-17.

```

<label class="blockLabel">
  Phone<span>*</span>
  <input type="text" id="phone" name="phone" />
</label>
<label class="blockLabel">
  Street Address<span>*</span>
  <input type="text" id="street" name="street" />
</label>

```

Adding the phone and street fields

Figure 6-17

```

<fieldset id="contact">
  <legend>Contact Information</legend>
  <label class="blockLabel">
    First Name<span>*</span>
    <input type="text" id="firstName" name="firstName" />
  </label>
  <label class="blockLabel">
    Last Name<span>*</span>
    <input type="text" id="lastName" name="lastName" />
  </label>
  <label class="blockLabel">
    Phone<span>*</span>
    <input type="text" id="phone" name="phone" />
  </label>
  <label class="blockLabel">
    Street Address<span>*</span>
    <input type="text" id="street" name="street" />
  </label>
</fieldset>

```

3. Save your changes to the file and then reload the donations page in your browser. Verify that the two additional input boxes have been added to the page in the same style and layout as the First Name and Last Name boxes, as shown in Figure 6-18.

Figure 6-18

Required contact information in the donations form

The next fields in the **Contact Information** field set are the city, state, and zip fields. Terry has indicated that she wants these three fields to be displayed on the same line in the form, just as they usually appear in mailing addresses. You'll add the labels for these control elements without the `blockLabel` class attribute so that the browser treats them as inline elements; however, you'll indent the first label for the city field by 150 pixels, lining it up with the rest of the columns in the form. You learn from Terry that the city and state fields are required by the CGI script that will process this form, so you'll add red asterisks to those two labels.

To add the city, state, and zip fields:

1. Return to the `donations.htm` file in your text editor.
2. Add the following elements to the form, as shown in Figure 6-19.

```
<label class="indentLabel">
  City<span>*</span>
  <input type="text" id="city" name="city" />
</label>
<label>
  State<span>*</span>
  <input type="text" id="state" name="state" />
</label>
<label>
  ZIP
  <input type="text" id="zip" name="zip" />
</label>
```

Figure 6-19

Adding the city, state, and zip fields

```
<label class="blockLabel">
  Street Address<span>*</span>
  <input type="text" id="street" name="street" />
</label>
<label class="indentLabel">
  City<span>*</span>
  <input type="text" id="city" name="city" />
</label>
<label>
  State<span>*</span>
  <input type="text" id="state" name="state" />
</label>
<label>
  ZIP
  <input type="text" id="zip" />
</label>
</fieldset>
```


- ▶ 3. Save your changes to the file and then return to the **forms.css** file in your text editor to create a style for the **indentLabel** class.
- ▶ 4. Add the following style to the bottom of the style sheet:
`label.indentLabel {margin-left: 150px}`
- ▶ 5. Save your changes to the style sheet and then reload the **donations.htm** file in your Web browser. Figure 6-20 shows the current layout of the form.

Form layout for the city, state, and zip fields

Figure 6-20

Contact Information

First Name *

Last Name *

Phone *

Street Address *

City * State * ZIP *

input box for the zip field wraps to a new line

Trouble? Under some browsers such as Safari, the three input boxes will not wrap onto a new line but will instead be displayed on a single line, crossing the boundary of the field set box.

The three input boxes for the city, state, and zip fields do not fit onto a single line, causing the input box for the zip field to wrap onto a second line. By default, most browsers set the width of the input boxes to display about 20 characters of text at any one time. You can change the width of these input boxes using the CSS width style.

Setting the Width of an Input Box

Because Terry wants users to enter only a two-letter abbreviation for the state input box, you can reduce the width of that box to 3 em. The width of the zip code input box can be reduced to 7 em. Finally, she would like the width of the city and phone input boxes set at 10 em. Terry thinks the other input boxes could be wider and suggests that you set the width of the firstName, lastName, and street input boxes to 25 em. Add these styles to the forms.css style sheet.

To set the width of the input boxes:

- ▶ 1. Return to the **forms.css** style sheet and add the following styles to the bottom of the sheet, as shown in Figure 6-21.

```
#firstName, #lastName, #street {width: 25em}
#phone, #city                {width: 10em}
#state                       {width: 3em}
#zip                         {width: 7em}
```

Figure 6-21

Setting the width of the input boxes

```
#donationForm span      {color: red;}
label.indentLabel      {margin-left: 150px}

#firstName, #lastName, #street {width: 25em}
#phone, #city           {width: 10em}
#state                  {width: 3em}
#zip                    {width: 7em}
```

2. Save your changes to the style sheet and reload the **donations.htm** file in your Web browser. Figure 6-22 shows the layout of the form with the new widths for the input boxes.

Figure 6-22

Input boxes with modified widths

Contact Information

First Name*

Last Name*

Phone*

Street Address*

City* State* ZIP

Applying new widths to the different input boxes has removed the line wrap from the form and made the form easier to read. The width style is one way of setting the size of an input box. For older browsers, you can also apply the size attribute to the input element as follows

```
<input type="text" size="chars" />
```

where chars is the number of characters displayed in the input box. For example, the tag

```
<input type="text" id="zip" name="zip" size="7" />
```

sets the width of the input box for the zip field to seven characters. This is not an exact measure because the width of individual characters varies (unless you specify a monospace font for the input box text).

Setting the Maximum Width of an Input Box

Setting the width of an input box does not limit the number of characters the box can hold. If a user tries to enter text longer than a box's width, the text scrolls to the left, hiding the extra characters. A user would not be able to see the entire text entered into the input box, but all of it would still be sent to the server for processing.

There are times when you want to limit the number of characters a user can enter in order to reduce the chance of erroneous data entry. For example, if you have a Social Security Number field, you know that only nine characters are required and that any

attempt to enter more than nine characters would indicate a mistake. To set the maximum number of characters allowed for an input box, you add the attribute

```
<input type="text" maxlength="chars" />
```

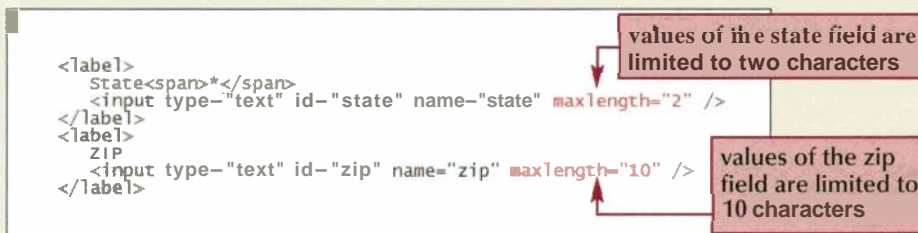
to the input element, where *chars* is the maximum number of characters that can be stored in the field. For the donation form, Terry wants users to enter the two-letter state abbreviation, so she suggests that you limit the size of that input box to two characters. She also wants to limit the width of the zip code field to 10 characters, allowing users to enter a nine-digit zip code that incorporates a hyphen.

To set the maximum width of the state and zip input fields:

1. Return to the **donations.htm** file in your text editor and add the attribute `maxlength="2"` to the input box for the state field.
2. Add the attribute `maxlength="10"` to the input box for the zip field. Figure 6-23 shows the revised code.

Setting the maximum number of characters for an input box

Figure 6-23



3. Save your changes to the file and reload **donations.htm** in your Web browser.
4. Click the input boxes for the state and zip fields, and verify that you cannot type more than two characters into the state field and more than 10 characters into the zip field.

Setting a Default Value for a Field

If you expect that most people will enter the same value into a field in your form, it might make sense to define a default value for that field. This makes data entry easier for users who want that default value, and it increases the accuracy of data entered into your Web form. To define a default value, you add the `value` attribute

```
<input value="value" />
```

to the input control element, where *value* is the default text or number that is displayed in the field. In the case of an input box, the default value appears in the input box when the form is initially opened.

Reference Window | Working with Input Box Attributes

- To set the size of the input box in characters, add the attribute `size="chars"` to the input element, where `chars` is the number of characters displayed in the input box.
- To set the maximum number of characters in the input box, use the attribute `maxlength="chars"` where `chars` is the maximum number of characters that can be entered into the input box.
- To set the default value of the field in the input box, use the attribute `value="value"` where `value` is the default value that will appear in the input box when the form is initially displayed.

About 80% of the online donations to The Lighthouse come from donors in St. Peters, Missouri. Terry suggests that you enter the city and state abbreviation into the form as a default value.

To set the default value for the `city` and `state` fields:

- ▶ 1. Return to the `donations.htm` file in your text editor and add the attribute `value="St. Peters"` to the input box for the city field.
- ▶ 2. Add the attribute `value="MO"` to the input box for state field, as shown in Figure 6-24.

Figure 6-24

Defining a default city and state value

```
<label class="indentLabel">
  City<span></span>
  <input type="text" id="city" name="city" value="St. Peters" />
</label>
<label>
  State<span></span>
  <input type="text" id="state" name="state" maxlength="2" value="MO" />
</label>
```

St. Peters is the default value for the city field

MO is the default value for the state field

- ▶ 3. Save your changes to the `donations.htm` file and then reload it in your browser. As shown in Figure 6-25, the default values of St. Peters and MO appear in the city and state fields, respectively.

Web form with default city and state values

Figure 6-25

Contact Information

First Name*

Last Name*

Phone*

Street Address*

City* St. Peters State* MO ZIP

- If you want to take a break before starting the next session, you can close your files and programs now.

Note that if donors from places other than St. Peters, Missouri use this Web form, they can remove the default value by selecting the text and pressing the Delete key.

Navigating Forms with Access Keys

InSight

In this session, you've activated control elements either by using your mouse button or by tabbing from one control element to another. As your forms get larger with more elements, you might want to give users the ability to jump to a particular element in the form. This can be done with an access key. An **access key** is a single key on the keyboard that you type in conjunction with the Alt key for Windows users, or the Control key for Macintosh users, to jump to one of the control elements in the form. You can create an access key by adding the `accesskey` attribute to any of the control elements discussed in this tutorial. For example, to create an access key for the `lastName` field, enter the following code:

```
<input type="text" name="lastName" id="lastName" accesskey="l" />
```

If a user types **Alt+l** (or **Command+l** for Macintosh users), the input box for the `lastName` field is selected. Note that you must use letters that are not reserved by your browser. For example, **Alt+f** is used by many browsers including Internet Explorer to access the File menu. If you use an access key, you should provide some visual clues about the key's existence. The accepted method is to underline the character corresponding to the access key. For example, in the previous code, you might display the Last Name label as Last Name.

You've completed the text input boxes for the Contact Information section of the donation form. In the next session, you'll complete the layout of the form by adding new fields to the form, including option buttons, selection lists, and check boxes.

Session 6.1 Quick Check

Review

- What is a CGI script?
- Specify the code to create a form with the name registration.
- Specify the code to create a field set with the id `contactInfo` and the legend Contact Information.
- What are two ways of associating a field label with a control element?
- Specify the code to create a field label with the text Phone that is associated with an input box containing the phone field.

6. What attribute would you add to the Phone input box to allow no more than 10 characters to be entered?
7. Specify the code to create an input box named subscribe with a default value of Yes.
8. What style would you enter to display all text input boxes as block-level elements?

Session 6.2

Creating Option Buttons

Donations to The Lighthouse come from both private individuals and businesses. Terry handles the receipts and thank you notes for private donations differently than those for business donations, so she would like the form to indicate whether the contact information is associated with a business or represents a home address. Terry doesn't want donors to enter this information in an input box; she would prefer that they enter the information with option buttons.

Option buttons, also called radio buttons, allow users to select a data value from a limited set of possible values. With option buttons, users can select only one button at a time from a group. The syntax to create a collection of option buttons is

```
<input type="radio" name="name" id="id1" value="value1" />
<input type="radio" name="name" id="id2" value="value2" />
<input type="radio" name="name" id="id3" value="value3" />
...
```

where *name* identifies the field associated with the collection of option buttons; *id1*, *id2*, *id3*, etc. identify the specific options; and *value1*, *value2*, *value3*, etc. are the field values associated with each option. Notice that all options within the group have the same name value. In fact, the *id* attribute is required only if you intend to use a field label with the option button or need some way of distinguishing one option button from another for use with a program or script.

When a group of option buttons share the same name, this puts them in a group—so that selecting one option button automatically deselects all of the others. Figure 6-26 shows an example of a Web form that uses an option button group to indicate political party affiliations.

Figure 6-26

Creating a group of option buttons

```
<fieldset>
  <legend>Party Affiliation</legend>
  <label for="demoption">Democrat</label>
  <input type="radio" name="party" id="demoption" value="dem" />
  <label for="gopoption">Republican</label>
  <input type="radio" name="party" id="gopoption" value="gop" />
  <label for="indoption">Independent</label>
  <input type="radio" name="party" id="indoption" value="id" />
</fieldset>
```

HTML code

Party Affiliation

Democrat Republican Independent

open buttons

In this sample code, all of the option buttons have the field name party but each has a different value. Because they share the same name, a user can select only one of the option buttons. The field set box provides a visual clue that all of these option buttons are part of the same field, but the field set is only there to aid in the form's appearance—it is not part of the option button syntax.

By default, an option button is unselected; but you can set an option button to be selected by adding the checked attribute to the input element:

```
<input type="radio" checked="checked" />
```

In older Web pages, you might see this code also entered as

```
<input type="radio" checked />
```

with no value provided for the checked attribute. However, this format is not supported in the official specifications for HTML and XHTML and should be avoided in new Web pages.

Tip

Use **option** buttons when a field has a small number of possible values, of which the user can **select** only **one**; otherwise, we a selection list.

Creating a Group of Option Buttons

Reference Window

To create a group of option buttons associated with a single field, add the elements

```
<input type="radio" name="name" id="id1" value="value1" />
<input type="radio" name="name" id="id2" value="value2" />
<input type="radio" name="name" id="id3" value="value3" />
```

to the Web form, where *name* identifies the field associated with the collection of option buttons; *id1*, *id2*, *id3*, etc. identify the specific options; and *value1*, *value2*, *value3*, etc. are the field values associated with each option.

To specify the default option, add the following attribute to the `<input>` tag:

```
checked="checked"
```

Terry wants you to insert two option buttons at the top of the Contact Information field set with the labels Home and Business. The field name you'll use for this group of option buttons is `addressType`. To make it clear to donors that the two option buttons are related, you'll enclose them in a field set box.

To create the option buttons for the `addressType` field:

1. Reopen the `donations.htm` file in your text editor.
2. Directly below the Contact Information legend, insert the following field set containing two option buttons with associated field labels:

```
<fieldset id="addressOptions">
  <legend>Address For</legend>

  <label for="homeType">Home</label>
  <input type="radio" id="homeType" name="addressType"
    value="home" />

  <label for="busType">Business</label>
  <input type="radio" id="busType" name="addressType"
    value="business" />

</fieldset>
```

Figure 6-27 shows the revised code.

Tip

Always enclose your option button group within a field set box to provide a visual indication that the option buttons belong to the same field.

Figure 6-27

Inserting a field set containing an option button group

```

<form name="donationForm" id="donationForm">
  <fieldset id="contact">
    <legend>Contact Information</legend>
    <fieldset id="addressOptions">
      <legend>Address For</legend>
      <label for="homeType">Home</label>
      <input type="radio" id="homeType" name="addressType" value="home" />
      <label for="bustype">Business</label>
      <input type="radio" id="bustype" name="addressType" value="business" />
    </fieldset>
    <label class="blockLabel">
      First Name<span>*</span>
      <input type="text" id="firstName" name="firstName" />
    </label>
  </fieldset>

```

- 3 Save your changes to the **donations.htm** file and then reload it in your browser. Figure 6-28 shows the new field set containing the two option buttons from the **addressType** field.

Figure 6-28

Option buttons in the donations form

- 4 Test the option buttons by clicking each one, verifying that you can **select** only one option at a time. **Also** verify that you can select an option button by clicking the field label associated with the button,

You decide that the option button group would look better if it weren't as wide and if it were lined up with the other control elements in the form. You'll add this code to the style sheet.

To change the appearance of the option group:

1. Reopen the **forms.css** file in your text editor.
2. The **fieldset** element containing the option buttons has the id **addressOptions**. Add the following style to the bottom of the style sheet, as shown in Figure 6-29.

```
#addressOptions {width: 180px; margin-left: 150px}
```


Setting the format of the addressOptions field set

Figure 6-29

```
#firstName, #lastName, #street {width: 25em}
#phone, #city                 {width: 10em}
#state                        {width: 3em}
#zip                           {width: 7em}
#addressoptions               {width: 180px; margin-left: 150px}
```

- 3 Save your changes to the style sheet and reload the **donations.htm** file in your Web browser. Figure 6-30 shows the new appearance of the control elements in the form.

Revised format of the addressOptions field set

Figure 6-30

You've now entered all of the control elements for the Contact Information part of the form. Next you'll add fields that store the amount of the donation and the method of payment. You'll start by creating an input box in which donors enter the amount of their donation. You'll name this new field amount. You'll use the same blockLabel class style you used earlier in the form to format the input box and field label.

To insert an input box for the amount of the donation:

1. Return to the **donations.htm** file in your text editor.
2. Scroll down the file. Within the Donation Information field set, insert the following code, as shown in Figure 6-31.

```
<label class="blockLabel">
  Donation Amount<span>*</span>
  <input type="text" id="amount" name="amount" />
</label>
```

Adding an input box for the donation amount

```
<fieldset id="donation">
  <legend>Donation Information</legend>
  <label class="blockLabel">
    Donation Amount<span>*</span>
    <input type="text" id="amount" name="amount" />
  </label>
</fieldset>
```

- 3 Save your changes to the file and then reload **donations.htm** in your Web browser. Figure 6-32 shows the new donation amount input box.

Figure 6-32 Donation amount input box

The screenshot shows a web form titled "Information" with the following fields and options:

- Address For:** Home (selected) Business (selected)
- First Name***: Text input field
- Last Name***: Text input field
- Phone***: Text input field
- Street Address***: Text input field
- City***: St Peters
- State***: MO
- ZIP**: Text input field
- Donation Information**: Section header
- Donation Amount***: Text input field

In the next field in the donation form, donors are asked to specify the credit card type. To insert this information you'll use a selection list.

Creating a Selection List

A **selection list** is a list box that presents users with a group of possible field values. A selection list fulfills the same role as a group of option buttons and is used in situations where there are too many options to be easily listed on the form with option buttons. As with option buttons, selection lists help prevent spelling mistakes and erroneous data entries that can occur with text input boxes. A selection list is created using the elements

```
<select name="name" id="id">
  <option value="value1">text1</option>
  <option value="value2">text2</option>
  ...
</select>
```

where *name* and *id* identify the field associated with the selection list; *value1*, *value2*, etc. are the possible field values; and *text1*, *text2*, etc. are the entries in the selection list. The text entries are displayed to the user, while CGI scripts retrieving data from a selection list will often work with either the field value or the text entry. Figure 6-33 shows a selection list version of the party affiliation field described earlier with option buttons.

Figure 6-33 Creating a selection list

```

<select id="party" name="party">
  <option value="dem">Democrat</option>
  <option value="gop">Republican</option>
  <option value="ind">Independent</option>
</select>
```

HTML code

Democrat ▾
Democrat
Republican
Independent

rendered selection list

options appear when you click the arrow

Although the first text entry is displayed in a selection list, this is not a default value for the list. To specify which of the options should be selected by default, add the following `selected` attribute to the option element:

```
<option selected="selected" value="value">text</option>
```

In older code, you might also see the `selected` attribute entered without an attribute value, appearing as

```
<option selected value="value">text</option>
```

but this is considered poor syntax and is rejected in XHTML documents.

Grouping Selection Options

In a selection list, the options are presented in the same order as they appear in the HTML code. In long selection lists it might be difficult for users to locate a particular option value. You can organize the selection list options by placing them in **option groups** using the `optgroup` element

```
<select>
  <optgroup label="label1">
    <option>text1</option>
    <option>text2</option>
  ..
  </optgroup>
  <optgroup label="label2">
    <option>text1</option>
    <option>text2</option>
  </optgroup>
</select>
```

where *label1*, *label2*, and so forth are the labels for the different groups of options. The text for the label appears in the selection list above each group of items but is not a selectable item from the list. Figure 6-34 shows an example of a selection list in which the options are divided into two groups.

Figure 6-34

Organizing a selection list with option groups

```

<label for="candidate">Candidate</label>
<select id="candidate" name="candidate">
  <optgroup label="Democrat">
    <option>Tim Harris</option>
    <option>Gary Nielsen</option>
    <option>Kate Paulenty</option>
  </optgroup>
  <optgroup label="Republican">
    <option>Barbara Alt</option>
    <option>Peter Trudeau</option>
    <option>Maria Sandoval</option>
  </optgroup>
</select>

```

HTML code

selection list option group

The appearance of the option group label is determined by the browser. You can apply a style to an entire option group including its label, but there is no CSS style to change the appearance of the option group label alone.

Setting the Selection List Size

By default, selection lists display only the currently selected option value. You can change the number of options displayed by applying the size attribute

```
<select size="value"> ... </select>
```

to the select element, where value is the number of items that the selection list displays in the form at a time. By specifying a value greater than 1, you change the selection list from a drop-down list box to a list box with a scroll bar that allows a user to scroll through the selection options. If you set the size attribute to be equal to the number of options in the selection list, the scroll bar either is not displayed or is dimmed, as shown in Figure 6-35.

Figure 6-35

Setting the size of the selection list

Candidate Tim Harris size="1" Candidate Tim Harris
Gary Nielsen
Kate Paulenty
Barbara Alt size="4" Candidate Tim Harris
Gary Nielsen
Kate Paulenty
Barbara Alt
Peter Trudeau
Maria Sandoval size="6" (all)

Although the `size` attribute defines the number of options displayed in the list box, there is no HTML attribute to set the width of the list box. The browser will make the width large enough to display the longest option text. If you want to change the width of a list box, you can use the CSS `width` style.

Creating a Selection List

Reference Window

- To create a selection list, add the elements

```
<select name="name" id="id">
  <option value="value1">text1</option>
  <option value="value2">text2</option>
```

```
  ...
```

```
</select>
```

to the Web form, where *name* and *id* identify the field associated with the selection list; *value1*, *value2*, etc. are the possible field values; and *text1*, *text2*, etc. are the entries in the selection list.

- To specify the default option, add the following attribute to the `<option>` tag:

```
selected="selected"
```

To set the number of options displayed at one time in the selection list, add the attribute

```
size="value"
```

to the `<select>` tag, where *value* is the number of options displayed in the selection list at a time.

Now that you've learned about selection lists, you can add one to the donation form for entering the credit card brand. The Lighthouse accepts payments from American Express, Discover, MasterCard, and Visa. The field values Terry wants you to use for these three vendors are Amex, Disc, MC, and Visa, respectively. She wants the values to be stored in a field named `creditCard`. The code for the selection is:

```
<select id="creditCard" name="creditCard">
  <option value="Amex">American Express</option>
  <option value="Disc">Discover</option>
  <option value="MC">MasterCard</option>
  <option value="Visa">Visa</option>
</select>
```

Terry wants the selection list displayed as a block-level element, with the field label placed alongside it.

To create a selection list for the credit card vendors:

- Return to the `donations.htm` file in your text editor.
- Below the donation amount input box, insert the following code, as shown in Figure 6-36.

```
<label class="blockLabel">
  Credit Card<span>*</span>
  <select id="creditCard" name="creditCard">
    <option value="Amex">American Express</option>
    <option value="Disc">Discover</option>
    <option value="MC">MasterCard</option>
    <option value="Visa">Visa</option>
  </select>
</label>
```

Figure 6-26

Inserting a selection list

```

<fieldset id="donation">
  <legend>Donation Information</legend>

  <label class="blockLabel">
    Donation Amount<span>*</span>
    <input type="text" id="amount" name="amount" />
  </label>

  <label class="blockLabel">
    Credit Card<span>*</span>
    <select id="creditCard" name="creditCard">
      <option value="Amex">American Express</option>
      <option value="Disc">Discover</option>
      <option value="MC">MasterCard</option>
      <option value="visa">visa</option>
    </select>
  </label>
</fieldset>

```

selection list
options

3. Save your changes to the file, and then return to the **forms.css** style sheet in your text editor. Like the input boxes you created earlier, you want the selection list positioned 150 pixels from the left margin of the field label. Add the following style to the bottom of the style sheet, as shown in Figure 6-37:

```
#creditCard {position: absolute; left: 150px}
```

Figure 6-37

Formatting the selection list

```

#firstName, #lastName, #street {width: 25em}
#phone, #city {width: 10em}
#state {width: 3em}
#zip {width: 7em}

#addressoptions {width: 180px; margin-left: 150px}
#creditCard {position: absolute; left: 150px}

```

4. Save your changes to the style sheet, and then reload **donations.htm** in your Web browser. Figure 6-38 shows the selection List for the **creditCard** field.

Figure 6-38

Credit card selection list in the donations form

Donation Information

Donation Amount*

Credit Card*

5. Click the selection list control for the **creditCard** field and verify that it displays the names of the four credit cards accepted by The Lighthouse.

The next two fields in the donation form are the **cardHolder** and **cardNumber** fields, which will be input boxes for users to enter the name on the credit card and the credit card number. You'll format these elements using the **blockLabel** label class, setting the width of both input boxes to 25 em.

To create input boxes for the card holder name and the credit card number:

1. Return to the **donations.htm** file in your text editor
2. Below the selection list, insert the following control elements, as shown in Figure 6-39.

```
<label class="blockLabel">
  Cardholder Name<span>*</span>
  <input type="text" id="cardHolder" name="cardHolder" />
</label>

<label class="blockLabel">
  Card Number<span>*</span>
  <input type="text" id="cardNumber" name="cardNumber" />
</label>
```

Adding input boxes for the cardholder name and credit card number

Figure 6-39

```
<label class="blockLabel">
  Credit Card<span>*</span>
  <select id="creditCard" name="creditCard">
    <option value="Amex">American Express</option>
    <option value="Disc">Discover</option>
    <option value="MC">MasterCard</option>
    <option value="Visa">Visa</option>
  </select>
</label>

<label class="blockLabel">
  Cardholder Name<span>*</span>
  <input type="text" id="cardHolder" name="cardHolder" />
</label>

<label class="blockLabel">
  Card Number<span>*</span>
  <input type="text" id="cardNumber" name="cardNumber" />
</label>

</fieldset>
```

- 3 Save your changes to the file, and then return to the **forms.css** style sheet in your text editor. Add the following styles at the bottom of the file to set the width on the cardHolder and cardNumber input boxes:

```
#cardHolder, #cardNumber {width: 25em}
```

Save your changes to the style sheet, and then reload **donations.htm** in your Web browser. Figure 6-40 shows the input boxes for the cardHolder and cardNumber fields.

Cardholder Name and Card Number input boxes

Figure 6-40

The screenshot shows a portion of a web form titled "Donation Information". It includes several input fields: "Donation Amount" (a text box), "Credit Card" (a dropdown menu currently showing "American Express"), "Cardholder Name" (a text box), and "Card Number" (a text box). Each field is preceded by a red asterisk, indicating it is a required field.

The final credit card information you need to add to the form is the expiration date. You'll add two selection lists to collect this information. One selection list will contain the month values from January(01) to December (12). The other selection list will contain the year value, ranging from 2011 to 2015.

To create selection lists for the credit card expiration date

1. Return to the **donations.htm** file in your text editor.
2. Below the credit card number input box, insert the following code, as shown in Figure 6-41.

```
<label class="blockLabel">
  Expiration Date<span>*</span>
  <select id="expMonth" name="expMonth">
    <option value="01">January (01)</option>
    <option value="02">February (02)</option>
    <option value="03">March (03)</option>
    <option value="04">April (04)</option>
    <option value="05">May (05)</option>
    <option value="06">June (06)</option>
    <option value="07">July (07)</option>
    <option value="08">August (08)</option>
    <option value="09">September (09)</option>
    <option value="10">October (10)</option>
    <option value="11">November (11)</option>
    <option value="12">December (12)</option>
  </select>
  <select id="expYear" name="expYear">
    <option value="2011">2011</option>
    <option value="2012">2012</option>
    <option value="2013">2013</option>
    <option value="2014">2014</option>
    <option value="2015">2015</option>
  </select>
</label>
```

Figure 6-41

Creating selection lists for the expiration month and year

```
<label class="blockLabel">
  Card Number<span>*</span>
  <input type="text" id="cardNumber" name="cardNumber" />
</label>

<label class="blockLabel">
  Expiration Date<span>*</span>
  <select id="expMonth" name="expMonth">
    <option value="01">January (01)</option>
    <option value="02">February (02)</option>
    <option value="03">March (03)</option>
    <option value="04">April (04)</option>
    <option value="05">May (05)</option>
    <option value="06">June (06)</option>
    <option value="07">July (07)</option>
    <option value="08">August (08)</option>
    <option value="09">September (09)</option>
    <option value="10">October (10)</option>
    <option value="11">November (11)</option>
    <option value="12">December (12)</option>
  </select>
  <select id="expYear" name="expYear">
    <option value="2011">2011</option>
    <option value="2012">2012</option>
    <option value="2013">2013</option>
    <option value="2014">2014</option>
    <option value="2015">2015</option>
  </select>
</label>
</fieldset>
```

3. Save your changes to the file.

You also have to add styles to the **forms.css** style sheet to line up the **expMonth** and **expYear** selection lists with the other entries in the donation form.

4. Go to the **forms.css** file in your text editor and add the following styles to the bottom of the file, as shown in Figure 6-42.

```
#expMonth {position: absolute; left: 150px}
#expYear  {position: absolute; left: 280px}
```

```
#creditcard      (position: absolute; left: 150px)
#cardHolder, #cardNumber {width: 25em}
#expMonth        (position: absolute; left: 1
#expYear         (position: absolute; left:
```

Figure 6-42

5. Save your changes to the style sheet and reload **donations.htm** in your Web browser. Figure 6-43 shows all of the control elements that collect credit card data.

Control elements collecting credit card data

Figure 6-43

Donation Information

Donation Amount*

~~~~ € c a r American Express ▾

Cardholder Name\*

Card Number\*  2011 ▾

Expiration Date\* January (01) ▾

6. Test the selection lists by clicking the selection list arrows to verify that all of the year and month options are present.

## Allowing for Multiple Selections

In the code you just entered for the donation form, donors were limited to a single option—a certain kind of credit card and a specific month and year for the expiration date. However, selection lists also allow for multiple selections by applying the following multiple attribute to the select element:

```
<select multiple="multiple"> . . . </select>
```

In older code, you might see the minimized version of this attribute, removing the attribute value as follows:

```
sselect multiples . . . </select>
```

However, as with the selected attribute, this is not correct HTML or XHTML syntax and so you should avoid using it.

There are two ways for users to select multiple items from a selection list. For noncontiguous selections, press and hold the Ctrl key (or the Command key on a Macintosh) while making the selections. For a contiguous selection, select the first item, press and hold the Shift key, and then select the last item in the range. This selects the two items as well as all the items between them.

If you decide to use a multiple selection list in a form, be aware that the form sends a name/value pair to the server for each option the user selects from the list. This requires the server-based program to be able to handle a single field with multiple values. Check and verify that your server-based programs are designed to handle this before using a multiple selection list. In most cases, you are better served using check boxes rather than selection lists with multiple values. You'll examine check boxes next because Terry wants donors to be able to indicate if they're interested in volunteering at The Lighthouse.

## Working with Check Boxes

You use a **check box** control in situations where you are checking for the presence or absence of something, such as whether or not a donor is interested in volunteering at the center. Check boxes are created using the input element with the type attribute set to checkbox, as follows:

```
<input type="checkbox" name="name" id="id" value="value" />
```

The value attribute contains the value of the field when the check box is checked. If no value is provided, the value On is used by default. For example, the following code creates a check box for determining whether the user is a member of the Democratic party:

```
<label for="dem">Democrat?</label>
<input type="checkbox" name="dem" id="dem" value="yes" />
```

If the check box is selected, the browser will submit the name/value pair of dem/yes to the CGI script running on the Web server. A name/value pair is only sent to the server when the check box is checked by the user. By default, check boxes are not selected. To make a check box selected by default, add the following checked attribute to the input element:

```
<input type="checkbox" checked="checked" />
```

As with other form attributes, you will also see older code with this attribute used as:

```
<input type="checkbox" checked />
```

But once again, you should always provide an attribute value, even if most browsers accept this attribute without a value.

### Reference Window | Creating a Check Box

- To create a **check box**, add the element `<input type="checkbox" name="name" id="id" value="value" />` to the Web form, where *name* and *id* identify the check box field and *value* is the value of the check box field if the check box is selected.
- To specify that the check box is selected by default, add the following **attribute** to the `<input>` tag:  
`checked="checked"`

In the next section of the donation form, Terry wants you to add a few fields for recording customer feedback and comments. Terry wants donors to be able to select a check box indicating whether they're interested in doing volunteer work at The Lighthouse in addition to providing financial support. You'll insert this field with a check box control element.

### To create a check box for volunteers:

- Return to the **donations.htm** file in your text editor and go to the Feedback field set near the bottom of the file.
- Directly below the legend element, insert the following code, as shown in Figure 6-44.

```
<label>
  <input type="checkbox" id="volunteer" name="volunteer" />
  I'm interested in volunteering at The Lighthouse.
</label>
```

Adding a check box for the volunteer field

Figure 6-44

```
<fieldset id="feedback">
  <legend>Feedback</legend>
  <label>
    <input type="checkbox" id="volunteer" name="volunteer" />
    I'm interested in volunteering at The Lighthouse.
  </label>
</fieldset>
</form>
```

- Save your changes to the file, and reload **donations.htm** in your Web browser. Figure 6-45 shows the new check box control added to the Feedback field set box.

The volunteer check box control

Figure 6-45



Feedback  
 I'm interested in volunteering at The Lighthouse.

- Click the check box, and then click the field label to verify that you can alternately select and deselect the field with both the check box and its label.

Note that you did not specify a value for the volunteer field. When the form is eventually submitted to a CGI script, it will send the **name/value** pair as **volunteer/on** when the check box is selected in the form, which means the person would like Terry to contact him or her for volunteering. If the check box is not selected, no name/value pair will be sent, and Terry will not contact the person.

Typically, users navigate through a Web form using the Tab key, which moves the cursor from one field to another in the order that the field tags are entered into the HTML file.

You can specify an alternate order by adding the `tabindex` attribute to any control element in your form. When each element is assigned a tab index number, the cursor moves through the fields from the lowest index number to the highest. For example, to assign the tab index number 1 to the `firstName` field from the donation form, you would enter the following `tabindex` attribute to the control element:

```
<input name="firstName" id="firstName" tabindex="1" />
```

This code would ensure that the cursor is in the `firstName` field when the form is first opened. (Fields with zero or negative tab indexes are omitted from the tab order entirely.)

Web page designers can use tab index numbers in their forms without worrying about older browsers that do not support this new standard. Such browsers simply ignore the `tabindex` attribute and continue to tab to the fields in the order that they appear in the HTML file.

## Working with Text Area Controls

The final part of the Feedback field set includes a place where donors can offer comments about The Lighthouse. Because their comments might contain several lines of text, it would not be appropriate to enter those comments in an input box because input boxes are limited to a single line of text. Instead, you can create a control element that allows for extended text entries using the `textarea` element

```
ctextarea name="name" id="id">
  text
</textarea>
```

where `text` is default text that is placed in the text area box. You do not have to specify default text—this would leave the text area box empty on the form.

The size of the text area box is determined by the browser. Most browsers create a text area box that is about 20 characters long and two or three lines high. To change the dimensions of the text area box, you add the `rows` and `cols` attributes

```
ctextarea rows="value" cols="value"> ... </textarea>
```

where the `rows` attribute specifies the number of lines in the text area box and the `cols` attribute specifies the number of characters per line. You can also set the dimensions of the `textarea` element using the CSS `width` and `height` styles.

As you type text into a text area box, the text automatically wraps to a new line as it extends beyond the box's width. If more text is entered into the box than can be displayed, the browser automatically adds horizontal and vertical scroll bars to the box. You can control how the browser handles extra text by using the `wrap` attribute

```
ctextarea wrap="type"> ... </textarea>
```

where `type` is one of the values described in Figure 6-46.

### Tip

The `rows` and `cols` attributes are required under strict applications of XHTML.

Values of the wrap attribute

Figure 6-46

| Value | Description                                                                                                                                                                                                                                                                       |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | All the text is <b>displayed on a single line</b> , <b>scrolling</b> to the left if the text extends past the width of the box. Text goes to the next row in the box only if the Enter <b>key</b> is <b>pressed</b> . The text is sent to the CGI script in a single line.        |
| soft  | Text wraps automatically to the next line when it extends beyond the width of the input box. The text is still sent to the CGI script in a single line without any information about how the text was wrapped within the text area box.                                           |
| hard  | Text wraps automatically to the next line when it extends beyond the width of the input box. When the text is sent to the CGI script, the line-wrapping information is included, <b>allowing</b> the CGI script to work with the text exactly as it appears in the text area box. |

The wrap attribute is not part of the World Wide Web Consortium (W3C) specifications for HTML or XHTML, but all browsers support it. The default wrap value is soft, which allows the text to wrap automatically to a new line—note that this information is not sent to the CGI script. If you need to include the line wraps as part of the field value, use the following attribute value:

```
wrap="hard"
```

## Creating a Text Area Box

| Reference Window

- To create a text area box for multiple lines of text, use the `<textarea>` element

```
<textarea name="name" id="id">
  text
</textarea>
```

where `name` and `id` identify the field associated with the text area box and `text` is the default text that appears in the box.

- To specify the dimensions of the box, add the attributes

```
rows="value" cols="value"
```

to the `<textarea>` tag, where the `rows` attribute specifies the number of lines in the text area box and the `cols` attribute specifies the number of characters per line.

You decide to use a text area box for the donor comments, setting the size of the box to 50 characters wide by five lines high.

### To create the comments text area box:

- Return to the `donations.htm` file in your text editor.
- Below the volunteer check box, insert the following elements to create the text area box, as shown in Figure 6-47.

```
<label for="comments" class="blockLabel">Comments</label>
<textarea id="comments" name="comments"
  rows="5" cols="50">
</textarea>
```

Figure 6-47

## Adding a text area box

```

<fieldset id="feedback">
  <legend>Feedback</legend>
  <label>
    <input type="checkbox" id="volunteer" name="volunteer" />
    I'm interested in volunteering at The Lighthouse.
  </label>
  <label for="comments" class="blockLabel">Comments</label>
  <textarea id="comments" name="comments" rows="5" cols="50"></textarea>
</fieldset>

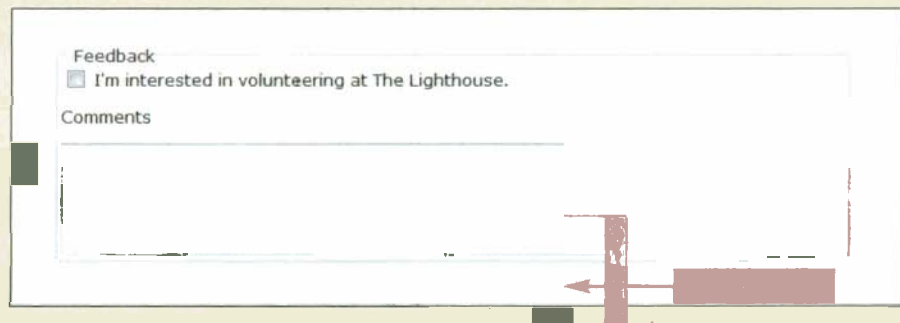
```

the text area box will have five lines of 50 characters each

3. Close the file, saving your changes, and then reload **donations.htm** in your Web browser. Figure 6-48 shows the comments text area box.

Figure 6-48

## Text area box in the Feedback section



4. Type some sample text into the text area box and verify that the text wraps to a new line as you exceed the width of the box.
- Trouble?** Line wraps do not occur in the middle of words. If you find your sample text is not wrapping to a new line, make sure you are entering individual words rather than a long character string.

You've added the last field to the donation form. Terry likes the layout of the form, but she thinks the field sets would look better if they had a light brown background similar to the color used in the page's background image. She also wants you to increase the space between and within each of the field set boxes.

## To format the field set boxes:

1. Go to the **forms.css** style sheet file in your text editor.
2. At the top of the style sheet, add the following style, as shown in Figure 6-49.

```

fieldset {margin-bottom: 10px; padding: 10px;
          background-color: rgb(237, 233, 223)}

```

Figure 6-49

## Setting the style of the fieldset elements

```

fieldset           {margin-bottom: 10px; padding: 10px; background-color: rgb(237,233,223)}
label.blockLabel  {display: block; position: relative; margin: 12px 0px}
label.blockLabel input {position: absolute; left: 150px}

```

- ▶ 3. Close the file, saving your changes, and then reload **donations.htm** in your Web browser. Figure 6-50 shows the revised design of the entire donation form.

Layout and design of the donation form

Figure 6-50

**Donation Form**  
\* indicates required information

**Contact Information**

Address For  
Home  Business

First Name\*

Last Name\*

Phone\*

Street Address\*

City\* ST Peters State\* MO ZIP

**Donation Information**

Donation Amount\*

Credit Card\* American Express ▾

Cardholder Name\*

Card Number\*

Expiration Date\* January (01) ▾ 2011 ▾

**Feedback**

I'm interested in volunteering at The Lighthouse.

Comments

**Trouble?** Depending on your browser, the tan background might not extend beyond the borders of the field set.

- ▶ 4. Take some time to work with all of the control elements you've entered, pressing the **Tab** key to move from one element to another and entering sample data into each field.
- ▶ 5. If you want to take a break before starting the next session, you can close all open files and programs now.

When you complete a Web form, you will probably want to ensure that the various form elements appear the same across different browsers and operating systems. The most natural way of doing this would appear to be with CSS. For example, you can apply the following style to display all command buttons with blue text on a yellow background:

```
input[type=button] {color: blue; background-color: yellow}
```

The degree to which these control elements can be formatted varies from browser to browser. Safari, for example, does not apply the above style, nor does it allow any changes to be made to any input element's border or background style. Other browsers are much more lenient. However, an important question for Web designers is whether to make these kinds of style changes even if the browser allows them.

One school of thought holds that any and all parts of a Web form should be open to CSS styles to enhance creativity in the design process. A different point of view holds that control elements such as input boxes and form buttons need to be, above all, usable. The most usable control element is one that has the same design and appearance as the other control elements found elsewhere on the user's computer. When users have come to expect a certain appearance for input boxes, command buttons, check boxes, and radio buttons, they can become confused by a Web format that has a totally different style.

However, the bottom line is that because the appearance of control elements is determined by the operating system and the browser, different browsers will apply CSS styles to control elements in different ways. Research has shown that trying to achieve a uniform look for control elements is a fruitless task. The best approach is to use CSS to lightly style form controls by **modifying** only properties such as font color, font size, and background colors--and realize that your style changes will not be seen by all of your users. As always, any styles applied to a Web form need to be checked under a variety of browsers and operating systems.

Terry likes the design and layout of the donations form. In the next session you'll add elements to the page to enable the donation form to interact with the CGI script running on The Lighthouse's Web server.

1. Specify the code to create two option buttons for the Computer field with the values PC and Macintosh.
2. In Question 1, what attribute would you add to the code to make PC the default value for the Computer field?
3. Specify the code to create a selection list for the State field with possible values of California, Nevada, Oregon, and Washington.
4. What attribute would you add to the code in Question 3 to make Oregon the default value for the State field?
5. In Question 3, what attribute would you add to the code to display all of the possible field values in the selection list?
6. How would you modify the code in Question 3 to allow for multiple selections?
7. Specify the code to create a check box and a label for the Computer field. The text of the label should be "I use a PC" and the value of the check box should be Yes.
8. Specify the code to create a text area box for the Memo field that displays 10 lines of text, each of which displays up to 40 characters.



## Session 6.3

### Working with Form Buttons

Up to now, all of your control elements have involved entering field values. Another type of control element is one that performs an action. In forms, this is usually done with **form buttons**, which perform one of three actions:

- run a command
- submit the form to the CGI script running on the server
- cancel the data entry done in the form

The first type of button you'll examine is the command button.

#### Creating a Command Button

A **command button** runs a command on the Web page. This command can be a call to a program running on a Web server or to a program installed within the Web browser. Command buttons are created using the `input` element

```
<input type="button" value="text" />
```

where *text* is the text that appears on the button. By itself, a command button performs no actions on a Web page. To create an action for a command button, you have to write a script or program that runs when the button is clicked. This can be done using a programming language such as JavaScript. Because that is beyond the scope of this tutorial, we won't examine how to use command buttons on the donation page.

#### Creating Submit and Reset Buttons

The two other kinds of form buttons are submit and reset buttons. A **submit button** submits a form to the server for processing when clicked. Clicking the **reset button** resets a form, changing all field values to their original default values and deleting any values that the user might have entered into the form. The syntax for creating these two buttons is

```
<input type="submit" value="text" />
<input type="reset" value="text" />
```

where the *value* attribute defines the text that appears on the button.

You can also specify `name` and `id` attributes for command, submit, and reset buttons, although these attributes are not required. You would use these attributes when a form contains multiple buttons and you're running a program that needs to distinguish one button from another. You won't need to add `id` and `name` attributes to the buttons on the donation form.

#### Tip

Avoid overpopulating your forms with buttons; too many buttons can be confusing. If more than **one** button is displayed, **use CSS styles** to give more visual emphasis on the button that will be most often clicked by the user.

Reference Window | **Creating Form Buttons**

- To create a form button to run a command, use the element  
`<input type="button" value="text" />`  
 where text is the text that appears on the button.
- To create a form button to submit the form and its fields and values to a CGI script, use the element  
`<input type="submit" value="text" />`  
 To create a form button to reset the form to its default values and appearance, use  
`<input type="reset" value="text" />`

Terry wants the donation form to include both a submit button and a reset button. The submit button, which she wants labeled Send Donation, will send the form data to the server for processing when clicked. The reset button, which she wants labeled Cancel, will erase the user's input and reset the fields to their default values.

**To add the submit and reset buttons to the donation form:**

- ▶ 1. Return to the **donations.htm** file in your text editor.
- ▶ 2. Scroll to the bottom of the file. Directly below the closing `</fieldset>` tag for the feedback field set, insert the following input elements, as shown in Figure 6-51:

```
<input type="submit" value="Submit Donation" />
<input type="reset" value="Cancel" />
```

**Figure 6-51** Adding submit and reset buttons

```
</fieldset>
<input type="submit" value="Submit Donation" />
<input type="reset" value="Cancel" />
</form>
</div>
<address>
  The Lighthouse &bull;
  150 Cavates Rd. &bull;
  St. Peters, MD 63376 &bull;
  (636) 555 - 4477
</address>
```

- ▶ 3. Save your changes to the file, and then reload **donations.htm** in your Web browser. Figure 6-52 shows the complete donations page with the Web form.

Completed donation page

Figure 6-52

The success of The Lighthouse reflects the dedication and support of members of the \* h o b make our dream a reality. We cannot continue to operate without contributions from people like you.

You can make a tax-deductible donation online using your American Express, Discover, MasterCard, or Visa card. Please fill out the form on this page.

The Lighthouse is always looking for volunteers. We especially need help in the following areas:

- Mechanics
- Carpenters
- Electricians
- Cooks
- Computer technicians
- Babysitters
- Data entry persons

and many others. Please consider donating your time and talents to your community and your neighbors.

Thank you so much for your generosity!

— Terry Ives  
Director, *The Lighthouse*

The Lighthouse • 150 Cavates Rd. • St. Peters, MO 63376 • (636) 555 - 4477

## Donation Form

\* indicates required information

Information

Address For  
Home  Business

First Name\*

Last Name\*

Phone\*

Street Address\*

City\* St. Peters State\* MO ZIP

Donation Information

Donation Amount\*

Credit Card\* American Express

Cardholder Name\*

Card Number\*

Expiration Date\* January (01)  2011

Feedback

I'm interested in volunteering at The Lighthouse.

Comments

Submit Donation Cancel

4. Test the Cancel button by entering data into the form and then clicking the Cancel button. Verify that the form is reset to its initial state and default values.

## Designing a Custom Button

The text of a command, submit, or reset button is determined by the value attribute. You are only allowed to specify the button label text. You can't add other elements such as an inline image to the button value. For more control over a form button's appearance you can use the button element

```
<button name="name" id="id" value="value" type="type">
  content
</button>
```

### Tip

Custom buttons help personalize a Web site and make it more user friendly.

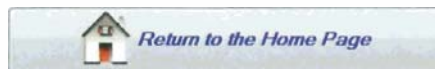
where the name and *value* attributes identify the button and the value sent to a server-based program, respectively; the *id* attribute specifies the button's id; the type attribute specifies the button type (submit, reset, or button); and content are page elements displayed within the button. The page content can include formatted text, inline images, and other design elements supported by HTML. Figure 6-53 shows an example of a button that contains both formatted text and an inline image.

Figure 6-53

### Creating a custom button

```
<button name="home" id="home" type="button">
  
  <span style="color: blue; font-weight: bold; font-style: italic">
    Return to the Home Page
  </span>
</button>
```

HTML code



custom button

## Creating File Buttons

Another type of button supported by HTML is the **file button**, which is used to select files so that their contents can be submitted for processing to a program. File buttons are created by applying the attribute

```
type="file"
```

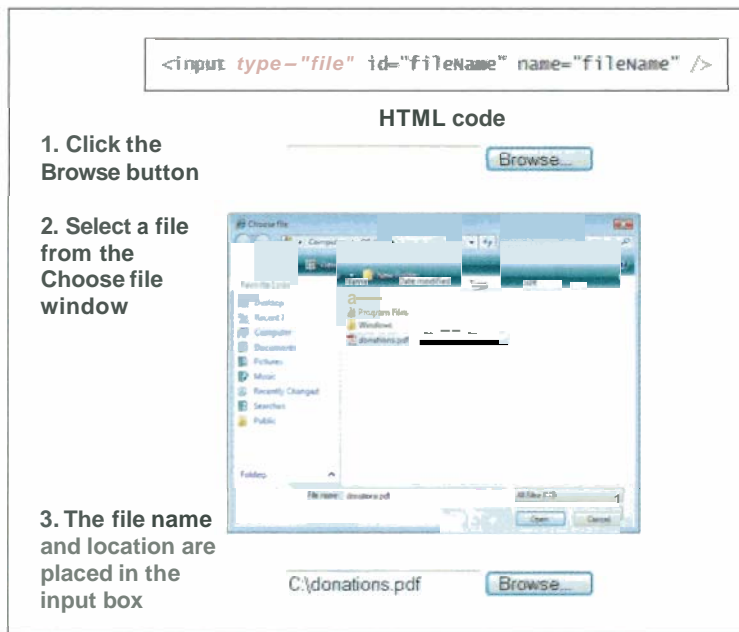
to the input element as follows:

```
<input type="file" name="name" id="id" />
```

Most browsers render file buttons as input boxes accompanied by a Browse button. As shown in Figure 6-54, when the user clicks the Browse button, a window opens from which the user can select a file. The file's location and name are then automatically inserted into the input box. When the form is submitted for processing, a script could use the value of the input box to retrieve the file as long as the Web server has access to the folder in which the file is stored.

Using a file button

Figure 6-54



You cannot change the label for the Browse button, but you can increase the size of the input box using either a CSS style or the HTML size attribute.

## Creating Image Field Buttons

Another control element you can use in your Web form is an image button. Image buttons act like submit buttons, allowing a user to click an image to submit a form. The syntax for this type of control element is

```
<input type="image" src="url" name="text" id="id" />
```

where *url* is the filename and location of the inline image. The user interacts with this control element by clicking somewhere within the image.

The image field can also act as an image map by recording where within the image the user clicked. When the form is submitted to a server-based program, the coordinates of that mouse click are attached to the image's name in the format

```
name.x=coordinate&name.y=coordinate
```

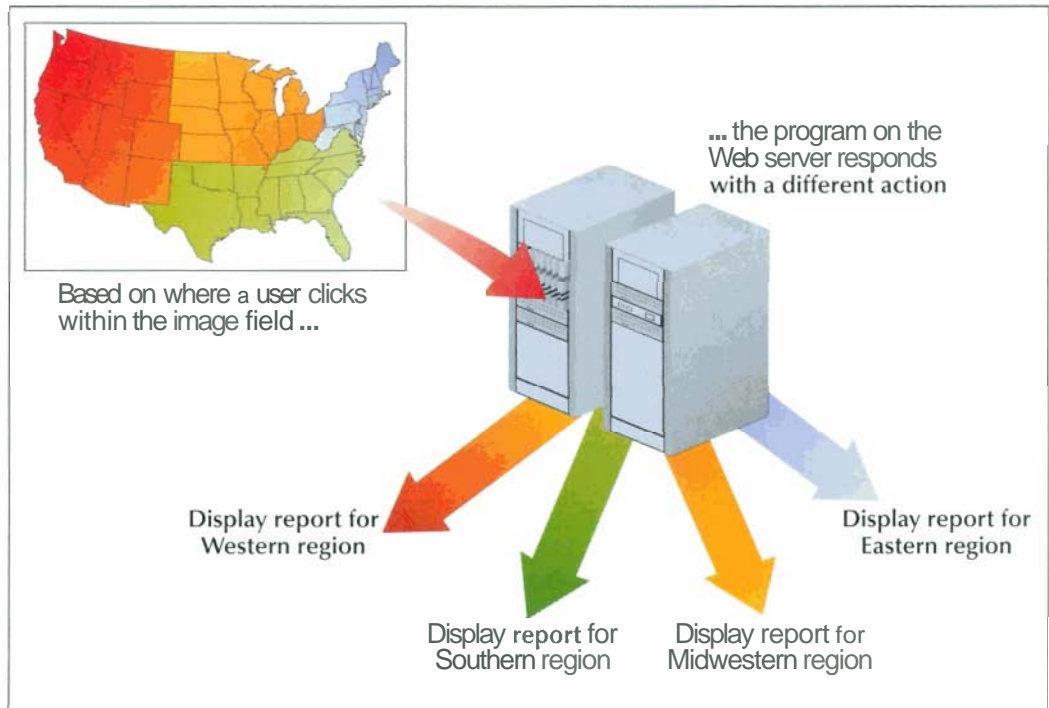
where *name* is the name of the image field and *coordinate* are the coordinates of the mouse click in the *x* and *y* direction. For example, suppose your Web page contains the following inline image form element:

```
cinput type="image" src="usamap.gif" name="usa" id="usa" />
```

If a user clicks the inline image at the coordinates (15, 30), the Web form sends the text string "usa.x=15&usa.y=30" to the server. Once the server-based program receives this data, it performs an action in response to that mouse click, as shown in Figure 6-55.

Figure 6-55

Using an image control field with a server-based program



Terry doesn't need any inline image controls or file buttons in the Web form for The Lighthouse.

## Working with Hidden Fields

Terry is pleased with the final appearance of the donation form. She shows the code for the form to Warren Kaughman, one of the programmers responsible for the CGI script that will process the donations. Warren notices only one thing missing from the code: the e-mail address that will receive and processes a new donation.

Unlike the other fields you've created so far, this field has a predefined value that users of the Web form should not be able to change. In fact, the e-mail address for donations should not even be displayed on the form. To accomplish this, you can use a **hidden field**, which is added to the form but not displayed on the Web page. The syntax for creating a hidden field is:

```
<input type="hidden" name="name" id="id" value="value" />
```

### Tip

Even though hidden fields are not displayed by the browser, their values can still be read by examining the source code; so do not put any sensitive information in a hidden field.

You've learned from Warren that the name of the e-mail field should be eMail; the e-mail address that will receive the new donations is *donations@thelighthouse.org* (note that this is a fictional address used for the purposes of this tutorial). Now that you know both the field name and the field value, you can add the hidden field to the donation form.

Because the field is hidden, you can place it anywhere within the form element. A common practice is to place all hidden fields in one location, usually at the beginning of the form, to make it easier to read and interpret your HTML code. You should also include a comment describing the purpose of the field.

### To add the hidden field to the donation form:

1. Return to the **donations.htm** file in your text editor.
2. Directly below the opening `<form>` tag, insert the following **element**, as shown in Figure 6-56.

```
<input type="hidden" name="eMail" id="eMail"
      value="donations@thelighthouse.org" />
```

Adding a hidden field

Figure 6-56

```
<form name="donationForm" id="donationForm">
  <input type="hidden" name="eMail" id="eMail"
        value="donations@thelighthouse.org" />
  <fieldset id="contact">
    <legend>Contact Information</legend>
```

3. Save your changes to the file.

## Creating a Hidden Field

## Reference Window

To create a hidden field, add the control element

```
<input type="hidden" name="name" id="id" value="value" />
```

to the form, where *value* is the value of the hidden field, and *name* and *id* identify the hidden field.

With the e-mail field now included in the donation form, you'll return to the first tag you entered into this document, the `<form>` tag, and insert the attributes needed for it to interact with the CGI script running on the organization's Web server.

## Working with Form Attributes

You've added all the elements needed for the form. Your final task is to specify where to send the form data and how to send it. You do this by adding the attributes

```
<form action="url" method="type" enctype="type"> ... </form>
```

to the form element, where *url* specifies the filename and location of the program that processes the form, the *method* attribute specifies how your Web browser sends data to the server, and the *enctype* attribute specifies the format of the data stored in the form's field. Next you'll examine the *method* and *enctype* attributes in more detail.

There are two possible values for the *method* attribute: *get* or *post*. The *get* method, the default, appends the form data to the end of the URL specified in the *action* attribute. The *post* method, on the other hand, sends form data in a separate data stream, allowing the Web server to receive the data through what is called **standard input**. Because it is more flexible, most Web designers prefer the *post* method for sending data to a server. Also, because browsers limit the size of URLs, the *post* method is safer—avoiding the possibility of data being truncated (this can happen using the *get* method if a long string is appended to a URL). The *post* method is also safer because the content of an extended URL can be viewed by other users and automated programs.

Don't be concerned if you don't completely understand the difference between get and post. Your Internet service provider can supply the necessary information about which of the two methods you should use when accessing the CGI scripts running on its server.

The `enctype` attribute determines how the form data should be encoded as it is sent to the server. Figure 6-57 describes the three most common encoding types.

**Figure 6-57** Values of the `enctype` attribute

| Value                                          | Description                                                                                                                                                                                                                                                                                                                              |
|------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>application/x-www-form-urlencoded</code> | The default format. In this format, form data is transferred as a long text string in which spaces are replaced with the <code>+</code> character and nontext characters (such as tabs and line breaks) are replaced with their hexadecimal code values. Field names are separated from their field values with a <code>=</code> symbol. |
| <code>multipart/form-data</code>               | Used when sending files to a server. In this format, spaces and nontext characters are preserved, and data elements are separated using delimiter lines. The action type of the form element must be set to post for this format.                                                                                                        |
| <code>text/plain</code>                        | Form data is transferred as plain text with no encoding of spaces or nontext characters. This format is most often used when the action type of the form element is set to <code>mailto</code> .                                                                                                                                         |

Finally, another attribute you might use with the form element is the `target` attribute, used to send form data to a different browser window or frame. This is not a concern with the donation form.

Now that you've been introduced to the issues involved in sending form data to a server-based program, you are ready to make some final modifications to the `donations.htm` file. Warren tells you that a CGI script that processes the form is located at the URL `http://www.thelighthouse.org/cgi-bin/donation` (a fictional address) and uses the `post` method. You do not have to specify a value for the `enctype` attribute, so the browser will assume a value of `application/x-www-form-urlencoded`.

#### To add attributes to the form element:

1. Return to the `donations.htm` file and add the following attributes to the opening `<form>` tag, as shown in Figure 6-58.

```
action="http://www.thelighthouse.org/cgi-bin/donation"
method="post"
```

**Figure 6-58** Setting the form attributes

```
<form name="donationForm" id="donationForm"
  action="http://www.thelighthouse.org/cgi-bin/donation"
  method="post">
  <input type="hidden" name="email" id="email"
    value="katherinehayes@thelighthouse.org" />
```

2. Close the `donations.htm` file, saving your changes.
3. You can close any other open files or programs.



## Using the mailto Action

The data from the donation form must be processed using a CGI script running on the center's Web server. There is, however, a way to send form information from the Web form to an e-mail address. You can do this with the `mailto` action, which accesses the user's own e-mail program and uses it to mail form information to a specified e-mail address, bypassing the need to use a CGI script. The syntax of the `mailto` action is

```
<form action="mailto:e-mail" method="post" enctype="text/plain"> ...
</form>
```

where *e-mail* is the e-mail address of the recipient of the form. Because the `mailto` action does not require a server-based program, you don't have to coordinate your form with a CGI script running on the Web server.

The `mailto` action is not supported by earlier versions of many browsers. Another concern is that using the `mailto` action requires the user filling out the form to have an e-mail client program that will be able to accept the output from the Web form and use it to send an e-mail message. This might not always be the case, as different users run different types of e-mail clients or might have no e-mail client at all. Finally, messages sent via the `mailto` action are not encrypted for privacy and therefore are a security risk. For these reasons, you should carefully consider all of the ramifications of the `mailto` action before using it in one of your forms. However, if these issues are not obstacles to your project, you can use the `mailto` action for situations where you need to send form data to an e-mail address and a CGI script is not available.

### Tips for Effective Forms

### InSight

Web forms are one of the main ways of communicating with your users; so it's important for the forms to be friendly and easy to use. A well-designed form can often be the difference between a new customer and a disgruntled user who leaves your site to go elsewhere. Here are some tips to remember when designing your form:

- Mark fields that are required, but also limit the number of unrequired fields. Don't overwhelm your users with requests for information that is not really essential. Keep your forms short and to the point.
- If you need to collect a lot of information, break the form into manageable sections spread out over several pages. Allow users to easily **move** backward and forward through the forms without losing data.
- Provide detailed instructions about what users are expected to do. Don't assume that your form is self-explanatory.
- If you ask for personal data and financial information, provide **clear** assurances that the data will be secure. If possible, provide a link to a Web page describing your security practices.
- Clearly indicate what users will receive once the form is submitted, and provide feedback on the Web site and through e-mail that tells them when their data has been successfully submitted.

Finally, every Web form should undergo usability testing before it is made available to the general public. Weed out any mistakes and difficulties before your users see the form.

You've finished the donation form, and Terry has placed a copy of `donation.htm` in a folder on the company's Web server. From there it can be fully tested to verify that the CGI script and the form work properly together. Terry is pleased with your work on this project and will come back to you for future Web page development at The Lighthouse.

1. Specify the code to create a submit button with the text Send Form.
2. Specify the code to create a reset button with the text Cancel Form.
3. Specify the code to create an image field named Sites displaying the graphic file sites.gif.
4. Specify the code to create a hidden field named Subject with the field value Form Responses.
5. You need your form to work with a CGI script located at *http://www.j\_davis.com/cgi-bin/post-query*. The Web server uses the get method. Specify the code for the form element.
6. You want to use the mailto action to send your form to the e-mail address *walker@j\_davis.com*. Assume that the message is sent as plain text. Specify the code for the form element.

In this tutorial, you learned how to create and use Web forms. The first session dealt with the fundamentals of Web forms, discussing how Web forms interact with the Web server to submit information to programs running on the server. You learned how to create and format simple input boxes with form labels, and you learned how to create field sets. You also saw how to use CSS styles to format the appearance and layout of a Web form. The second session examined other types of control elements, including option buttons, selection lists, and check boxes. The session concluded by examining how to create text area boxes for extended text input. The last session showed how to create form buttons for resetting a form or submitting it to a program-for processing. The session also examined some special input fields that can be used to create server-side image maps and file input boxes. The session and the tutorial concluded by examining various form attributes and discussed how data from the Web form is transferred to a CGI script running on a Web server.

### Key Terms

|                          |                 |                |
|--------------------------|-----------------|----------------|
| access key               | field set       | Perl           |
| CGI script               | field value     | post method    |
| check box                | form button     | radio button   |
| command button           | get method      | reset button   |
| Common Gateway Interface | hidden field    | selection list |
| script                   | input box       | standard input |
| control element          | name/value pair | submit button  |
| drop-down list box       | option button   | text area box  |
| field                    | option group    |                |

Practice | Review Assignments

Practice the skills you learned in the tutorial using the same case scenario.

Data Files needed for the Review Assignments: back.jpg, left.jpg, lhouse.jpg, main.css, right.jpg, vformstxt.css, and voltxt.htm

Terry and the staff at The Lighthouse have been working with your form and the CGI script running on the Web server for several weeks now. They're pleased with the work you've done, so they have asked for your help in creating another Web form for the center's Web site. Terry would like a form that Lighthouse volunteers can fill out indicating their talents and interests, and ways they can help the center. A CGI script is already in place to process the information, and much of the work in designing the volunteer page has been done except for the form itself. Terry wants you to complete the page by adding the HTML and CSS code for the volunteer form. A preview of the form you'll create is shown in Figure 6-59.

Figure 6-59

The screenshot shows a web browser window displaying the website for 'The Lighthouse'. The page has a header with the organization's name and logo, a navigation menu, and a main content area. A 'Volunteer Form' is overlaid on the page, containing two main sections: 'Contact Information' and 'Volunteer Information'. The form includes various input fields, checkboxes, and a dropdown menu. A footer at the bottom of the page provides contact information for The Lighthouse.

**The Lighthouse**  
 The Lighthouse 150 Cavates Rd. St. Peters, MO 63376

home projects upcoming events community links staff donations volunteers contact info

**Volunteer Form** \* indicates required information

Thank you for considering donating your time and talents to The Lighthouse. While we always welcome financial support, we cannot continue to serve the public without volunteers like you.

Please fill out the form on this page indicating your interests and talents. Everyone has something to offer, so don't be shy. And remember, we can always teach you, so don't let inexperience keep you away.

While you are helping others, you too will gain from the experience. Many of our volunteers have been with us for years. One of our volunteers told me the other day, "Terry, The Lighthouse has given more to me than I have ever given to it."

Welcome to the team!  
 — Terry Ives  
 Director, The Lighthouse

P.S. Please consider a financial donation to help us continue to support our neighbors in the community.

**Contact Information**

I am 16 or older\*

First Name\*

Last Name\*

Street Address 1\*

Street Address 2

City\*

State\*

ZIP

Phone\*

**Volunteer Information**

How did you hear about The Lighthouse?

Have you volunteered before?  
 Yes  No

I can help with the (check all that apply)

Baby Sitting  Cleaning  Clerical Duties

Event Planning  Mailing  Maintenance

Meal Preparation  Tutoring  Web Site

Tell us about yourself

The Lighthouse • 150 Cavates Rd. • St. Peters, MO 63376 • (636) 555 - 4477

When the form is filled out, it should be sent to a CGI script at <http://www.thelighthouse.org/volunteer>. The CGI script will collect the information and e-mail it to Steve Jones, the volunteer coordinator.

Complete the following:

1. Use your text editor to open the **voltxt.htm** and **vformstxt.css** files from the **tutorial.06\review** folder included with your Data Files. Enter your **name** and **the date** in the comment section of each file. Save the files as **volunteer.htm** and **vforms.css**, respectively, in the same folder.
2. Go to the **volunteer.htm** file in your text editor. Scroll down to the **rightColumn** div container and directly below the paragraph element, insert a form element with the name and id **volunteerForm**. Have the form perform the action of submitting the form data to the CGI script at <http://www.thelighthouse.org/cgi-bin/volunteer> using the post method.
3. Directly below the opening **<form>** tag, insert a hidden field named **eMail** with the value **stevejones@thelighthouse.org**.
4. Create a field set with the legend **Contact Information**. Give the fieldset element the id **contactFields**.
5. Below the field set legend, insert a label element with the text **"I am 16 or older\*"**. Enclose the asterisk symbol in a span element. Directly before the label text, but still nested within the label element, insert a check box with the field name **ageOK**.
6. Below the label element you just entered, insert the contact information for the volunteer. There are eight contact fields: **fName**, **lName**, **street1**, **street2**, **city**, **state**, **zip**, and **phone**. For each field, do the following:
  - Create an input box nested within a label element. The labels for the eight fields are **First Name\***, **Last Name\***, **Street Address 1\***, **Street Address 2**, **City\***, **State\***, **ZIP**, and **Phone\***.
  - Enclose the asterisks within a span element.
  - Place each label element within the **blockLabel** class.
7. Set the maximum number of characters in the state and zip fields to two and 10 characters, respectively.
8. Set the default value of the city and state fields to **St. Peters** and **MO**, respectively.
9. Below the **contactFields** field set, insert another field set with the legend **Volunteer Information**. Give the field set the id **volunteerInfo**.
10. Directly below the **Volunteer Information** legend, insert a selection list for the **infoSource** field. Add the following code for the selection list:
  - Enclose the selection list within a label element with the class **blockLabel**.
  - Directly before the selection list within the label element, insert the text **"How did you hear about The Lighthouse?"**
  - Add the following five options to the selection list: **Word of Mouth**, **TV or Radio Ad**, **The Internet**, **The Phonebook**, and **College/High School**. Give the five options the values **talk**, **ads**, **internet**, **phonebook**, and **schools**, respectively.

11. After the selection list label, insert a field set with the id `experience` and the legend "Have you volunteered before?" Within the field set, create two option buttons with the following code:
  - Before each option button, insert a label with the text strings "Yes" and "No". Use the `for` attribute to assign the labels to the `prevYes` and `prevNo` fields, respectively.
  - After each label, insert an option button belonging to the `prevExp` field. The ids of the option buttons should be `prevYes` and `prevNo`, respectively, and the values of the buttons should be `yes` and `no`.
12. After the `experience` field set, insert another field set with the id `interestFields` and the legend "I can help with the (check all that apply)".
13. Within the `interestFields` field set, insert nine check boxes. Format the check boxes as follows:
  - Enclose each check box within a label element. Give the label elements ids of `interest1` through `interest9`.
  - Give the check box controls the field names of `babysitting`, `cleaning`, `clerical`, `events`, `mailing`, `maintenance`, `food`, `tutoring`, and `web`.
  - After each check box within the label element, insert the text strings "Baby Sitting", "Cleaning", "Clerical Duties", "Event Planning", "Mailing", "Maintenance", "Meal Preparation", "Tutoring", and "Web Site".
14. Below the `interestFields` field set, insert a label associated with the comments field. Place the label in the `blockLabel` class and give it the text "Tell us about yourself".
15. After the label, insert a text area box for the comments field. The text area box should have five lines of 55 characters each.
16. After the text area box, insert submit and reset buttons. The text of the submit button should be "I'm Ready to Volunteer" and the text of the reset button should be "Cancel".
17. Go to the top of the file and link the file to the `vforms.css` style sheet.
18. Close the `volunteer.htm` file, saving your changes.
19. Go to the `vforms.css` file in your text editor and add the following styles to the style sheet:
  - Set the `background-color` of all `fieldset` elements to the value `(237,233,223)` with 10 pixels of padding and a bottom margin of 10 pixels.
  - Display all `span` elements within field sets in a red font.
  - Display all labels belonging to the `blockLabel` class as block-level elements, with relative positioning. Set the width of the labels to 450 pixels. Set the top and bottom margins to 12 pixels and the left and right margins to 0 pixels.
  - Place all input elements nested within `blockLabel` labels with absolute positioning 140 pixels to the left of the label's left margin.
  - Set the width of the `fName` and `lName` input boxes to 250 pixels. Set the width of the `street1` and `street2` input boxes to 350 pixels. Set the width of the phone and city input boxes to 150 pixels. Set the width of the state input box to 40 pixels and the width of the zip input box to 80 pixels.
  - Set the width of the `experience` field to 450 pixels with 5 pixels of padding.
20. Terry wants the nine check boxes that constitute different volunteer opportunities to be displayed in a grid of three rows and three columns. To create this layout:
  - Place the `interestFields` field set with relative positioning. Set the size of the field set box to 450 pixels wide by 120 pixels high. Set the padding to 5 pixels.
  - Place the `interest1` through `interest9` label elements with absolute positioning.

- Set the top coordinate of the interest1 through interest3 labels to 20 pixels, interest4 through interest6 to 50 pixels and interest7 through interest9 to 80 pixels.
  - Set the left coordinate of the interest1, interest4, and interest7 labels to 0 pixels; interest2, interest5, and interest8 to 140 pixels; and interest3, interest6, and interest9 to 280 pixels.
21. Save your edits to **vforms.css**, and then open **volunteer.htm** in your Web browser. Verify that the layout and design of the form resembles that shown in Figure 6-59.
  22. Submit your completed files to your instructor.

## Apply

### | Case Problem 1

Apply your knowledge of Web forms to create a subscription form for a newspaper.

**Data Files needed for this Case Problem:** **parch.jpg, pcg.css, pcglogo.jpg, sformtxt.css, and subtxt.htm**

**The Park City Gazette** Kevin Webber, the editor of the Park City Gazette of Estes Park, Colorado, has asked for your help in developing a subscription page for the newspaper's Web site. The page includes a form where customers can enter the length of the subscription they want to purchase, their mailing address, and their credit card information. Kevin has already created much of the layout and text of the Web page. Your job is to add the fields and control elements for the subscription form. A preview of the Web page you'll create for Kevin is shown in Figure 6-60.

Figure 6-60

The form contains several labels and control elements placed side-by-side in two columns. To create this two-column layout, you'll float the labels and control elements on the left margin. You'll identify the labels that are floated by putting them into the float-Label class. The floated control elements will belong to the floatCtrl class.

Complete the following:

1. Use your text editor to open the **sformtxt.css** and **subtxt.htm** files from the tutorial.06\case1 folder included with your Data Files. Enter **your name** and **the date** in the comment section of each file. Save the files as **sform.css** and **subscription.htm**, respectively, in the same folder.
2. Go to the **subscription.htm** file in your text editor and insert a link to the **sform.css** style sheet.
3. Scroll down the file and insert a form element with the id subForm, directly below the paragraph in the rightColumn div container.

4. At the top of the form, Kevin wants an option list showing the four different payment plans. Insert a field set with the id subPlans. Within the field set do the following;:-
  - Insert four option buttons belonging to the subplan field.
  - Give the option buttons the ids plan1 through plan4 and the values 1 through 4.
  - After each option button, insert a label element associated with the preceding option button. The text of the four labels is "6 mo./\$24", "12 mo./\$45", "18 mo./\$64", and "24 mo./\$80 (best value)".
5. After the subPlans field set, insert a label containing the text "Name". Associate the label with the cName field and put it in the class floatLabel.
6. After the label, insert an input box for the cName field. Place the input box in the floatCtrl field and set the size of the input box to 50 characters.
7. Insert another label containing the text "Address" associated with the address field and belonging to the floatLabel class. After the label, insert a text area box for the address field. Set the size of the box to six rows by 50 columns and place the text area box in the floatCtrl field.
8. Insert a label with the id agreeLabel associated with the agree field. Place the label in the floatLabel class. Within the label element, insert a check box for the agree field. After the check box, but within the label element, insert the text "Yes, I wish to pay online by entering my credit card information below."
9. Insert a field set with the id payment. At the top of the field set, insert a label belonging to the floatLabel class, containing the text "Credit Card" and associated with the cardType field.
10. Insert a selection list for the cardType field. Do the following for the selection list:
  - Place the selection list in the floatCtrl class.
  - Set the selection list to display four items.
  - Add the following options to the selection list: American Express, Discover, MasterCard, and Visa.
  - Set the values of the four options to: Amex, Disc, MC, and Visa.
11. Below the selection list, insert two labels. The first label should contain the text "Name on Card" and should be associated with the cardName field. The second label should contain the text "Card Number" and should be associated with the cardNumber field. Put both labels in the floatLabel class.
12. Directly after each label, insert an input box. The first input box should be for the cardName field; the second input box is for the cardNumber field. For both input boxes, set the width to 30 characters and place the input box into the floatCtrl class.
13. Insert a label belonging to the floatLabel class and containing the text "Expiration Date". After the label, insert two selection lists for the expMonth and expYear field. Do the following for the selection lists:
  - Place both selection lists in the floatCtrl class.
  - Add 12 options to the expMonth selection list containing the text "01" through "12". The values of the options should range from 1 to 12.
  - Add five options to the expYear selection list containing the text "2011" through "2015". Set the values of each option to match the option text.
14. Insert a field set with the id buttons. Within the field set, insert a submit and reset button. Give the submit button the value Subscribe. Give the reset button the value Cancel.
15. Use the CGI script at <http://www.theparkcitygazette.com/subscrib> with the post method.
16. Save your changes to the file.

 EXPLORE



17. Go to the **sform.css** file in your text editor and add the following styles to the style sheet:
  - Set the background color of the subPlans field set to white. Set the padding to 5 pixels and the bottom margin to 20 pixels. Center the contents of the field set.
  - For label elements within the subPlans fieldset element, set the right margin to 15 pixels.
  - Display objects belonging to the floatLabel class as block-level elements, floated on the left margin but only when the left margin is clear. (*Hint:* Use the clear style.) Set the width to 150 pixels and the bottom margin to 10 pixels. Right-align the label text.
  - Display objects belonging to the floatCtrl class as block-level elements, floated on the left margin. Set the left margin to 20 pixels and the bottom margin to 10 pixels.
  - Set the width of the agreeLabel label to 600 pixels with top/bottom margins of 20 pixels and left/right margins of 0 pixels. Center the label text.
  - Display the payment field set only when the left margin is clear. Set the background color to white. Set the width of the field set to 400 pixels with a left margin of 100 pixels and 5 pixels of padding.
  - Center the contents of the buttons field set with top/bottom margins of 10 pixels and left/right margins of 0 pixels. Set the border style to none.
18. Save your changes to the **sform.css** file and open **subscription.htm** in your Web browser. Verify that the layout and content of the Web form resemble that shown in Figure 6-60.
19. Submit your completed files to your instructor.

## Apply

## | Case Problem 2

*Apply your knowledge of Web forms to create a form for an online quiz.*

**Data Files needed for this Case Problem:** [cw.css](#), [cwlogo.gif](#), [cwquiztxt.htm](#), [qformtxt.css](#), and [tan.jpg](#)

**Civil War Studies** Adanya Lynne, a professor of military history at Ridgeview State College in Bartlett, Tennessee, has been preparing a series of online quizzes for her students. She has created the basic Web page design and layout, but has come to you for help in designing the quiz form. She envisions a series of multiple choice questions displayed in a collection of option buttons. Students will be able to click answers on the form and then submit their answers to a CGI script running on the Web server for their scores. Figure 6-61 shows a preview of the page you'll create for Professor Lynne.

Figure 6-61

**The Civil War**

## Quiz #1: Battles

Fill out the quiz and click the **Submit** button to see your score. Click the **Answers** button to see correct answers. Click the **Reset** button to reset the form.

|                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>Home</li> <li>Schedule</li> <li>Syllabus</li> <li>Course Notes</li> <li>Civil War Timeline</li> <li>Articles</li> <li>Online Quizzes</li> <li>Student Records</li> <li>Contact</li> <li>Information</li> </ul> | <p>1. Stonewall Jackson died in which battle?</p> <p>2. Chancellorsville pitted:</p> <p>3. Which Confederate general died at Shiloh?</p> <p>4. The "high water mark of the</p> <p>5. The Battle of Seven Days pitted:</p> <p>6. This battle involved trench warfare and foreshadowed WWI.</p> <p>7. The bloodiest day of the war occurred at:</p> <p>8. Shiloh pitted:</p> <p>9. Lost Confederate battle plans were an important precursor to:</p> <p>10. Which of these battles did not involve U.S. Grant?</p> | <p>a) Petersburg      b) Chancellorsville</p> <p>c) Fredricksburg      d) The Wilderness</p> <p>a) Lee vs. Hooker      b) Lee vs. Grant</p> <p>c) Lee vs. McClellan      d) Lee vs. Mead</p> <p>a) Stonewall Jackson      b) Joseph E. Johnston</p> <p>c) Albert Sidney Johnston      d) Jeb Stuart</p> <p>a) Antietam      b) Gettysburg</p> <p>c) Chancellorsville      d) Appomattox</p> <p>a) Lee vs. Grant      b) Lee vs. Hooker</p> <p>c) Johnston vs. Sherman      d) Lee vs. McClellan</p> <p>a) Gettysburg      b) Atlanta</p> <p>c) The Wilderness      d) Petersburg</p> <p>a) Antietam      b) Gettysburg</p> <p>c) Shiloh      d) Cold Harbor</p> <p>a) Lee vs. Hooker      b) Lee vs. Grant</p> <p>c) Johnston vs. Grant      d) Johnston vs. Sherman</p> <p>a) Gettysburg      b) Antietam</p> <p>c) Appomattox      d) Seven Days</p> <p>a) Gettysburg      b) Shiloh</p> <p>c) Petersburg      d) Vicksburg</p> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Complete the following:

1. Use your text editor to open the **cwquiztxt.htm** and **qformtxt.css** files from the `tutorial.06\case2` folder included with your Data Files. Enter **your name** and **the date** in the comment section of each file. Save the files as **cwquiz.htm** and **qform.css**, respectively.
2. Go to the **cwquiz.htm** file in your text editor and insert a link to the **qform.css** style sheet.
3. Scroll down the file to the `rightColumn` div element. Professor Lynne has inserted the 10 questions for the online quiz. Enclose the questions in a form element with the id `quizForm`. Call the CGI script at <http://www.ridgeviewcollmil.edu/quiz> using the post method.
4. Go to the first question and enclose the text of the question in a div element with the class name `question`.
5. Enclose the set of answers for the first question within a field set.

6. Place a label element around each possible answer for the first question. Put the first answer in the class `answerA`, the second answer in the class `answerB`, the third answer in the class `answerC`, and the fourth answer in the class `answerD`.
7. Within each of the four labels for the answers to the first question, insert an option button directly before the text of the answer. Assign the ids `ans1a`, `ans1b`, `ans1c`, and `ans1d` to the four option buttons. Assign the field name `question1` to each of the four option buttons. Finally, set the values for the four option buttons to `a`, `b`, `c`, and `d`.
8. Associate each of the four labels with a different id. Associate the first label with the `ans1a` field, the second with `ans1b`, the third with `ans1c`, and the fourth with `ans1d`.
9. Repeat Steps 4 through 8 for the remaining nine questions in the quiz, with the following changes:
  - Change the ids for the four option buttons to the question number. For example, the ids for the second question are: `ans2a`, `ans2b`, `ans2c`, and `ans2d`.
  - Change the field name of the four option buttons to the number of the question. For example, the field name for the second question is `question2`, for the third question is `question3`, and so forth.
10. After the last question, insert a `div` element with the id `buttons`. Within the element, insert a submit button with the value `Submit`, a command button with the value `Answers`, and a reset button with the value `Reset`.
11. Save your changes to the file.
12. Go to the `qform.css` file in your text editor and add the following styles to the style sheet:
  - Float all elements of the question class on the left margin. Set their widths to 200 pixels and right-align the text. Set the clear style to left so that the element is only displayed when the left margin is clear.
  - Display all field sets as block-level elements floated on the left margin. Set the size of the field sets to 400 pixels wide by 50 pixels high. Add a 20-pixel left margin and a 5-pixel bottom margin. Set the position property to relative. Finally, display the `tan.jpg` file as the background image for the field sets.
  - Use absolute positioning to place all elements of the `answerA` class at the coordinates (0, 0), place `answerB` class elements at the coordinates (200, 0), place `answerC` class elements at (0, 25), and place `answerD` class elements at (200, 25).
  - Center the contents of the `div` element with the `buttons` id. Set the width to 600 pixels and the top/bottom padding to 10 pixels. Set the left/right padding to 0 pixels.
  - For input elements within the `buttons` `div` element, set the width to 100 pixels, set the top/bottom margin to 0 pixels, and set the left/right margin to 10 pixels.
13. Save your changes to the file and open `cwquiz.htm` in your Web browser. Verify that the layout and design resemble that shown in Figure 6-61.
14. Submit your completed files to your instructor.

## Challenge | Case Problem 3

Explore different form controls needed to create an online order form for a pizzeria.

**Data Files needed for this Case Problem:** `buttonball.jpg`, `leftball.jpg`, `orderformtxt.css`, `pizzatxt.htm`, `rb.css`, `redball.jpg`, `rightball.jpg`, and `topping.txt`

**Red Ball Pizza** Alice Nichols is the owner of Red Ball Pizza, a new pizzeria in Ormond Beach, Florida. You've been working with Alice on creating a Web site for the restaurant. Alice wants to give customers the ability to submit orders online. She has contacted programmers at the restaurant's ISP to process the orders, but she needs a Web form to collect those orders. She's asked you to design a form that would allow customers to select items from the Red Ball Pizza menu. A preview of the Web page you'll create is shown in Figure 6-62.

Figure 6-62

**RED BALL PIZZA**

home page | menu | directions | coupons | online ordering | shopping cart | testimonials

**Online Ordering**

**Build Your Pizza**

Size: 14" Large

Crust: Hand Tossed

Toppings (select all that apply):

- Vegetables and Fruits
  - Diced mango
  - Artichokes
  - Broccoli
  - Caramelized onions

Quantity: 1

Add to Cart

**Extras**

Breadsticks: 6 piece

Cheesy Bread: 6 piece

Chicken Wings: 12 piece

Barbecue Wings: 12 piece

Add to Cart

Red Ball Pizza • 811 Beach Drive • Ormond Beach, FL 32175 • (386) 555 - 7499

Complete the following:

1. Use your text editor to open the `orderformtxt.css` and `pizzatxt.htm` files from the `tutorial.06\case3` folder included with your Data Files. Enter *your name* and *the date* in the comment section of each file. Save the files as `orderform.css` and `pizza.htm`, respectively.
2. Go to the `pizza.htm` file in your text editor and create a link to the `orderform.css` style sheet.
3. Scroll down the file to the `rightCol` div element and insert a form element directly below the `h1` heading.

**EXPLORE**

4. Create a field set with the id `pizzaType` and the legend Build your Pizza.
5. Insert a label element with the text Size. Within the label element, insert a selection list for the size field. Add the following options: 12" Regular, 14" Large, 16" Extra Large, and 20 Family Size. Set the option values to 12, 14, 16, and 20. Make 14" Large the default selection.
6. Insert a label element with the text "Crust". Within the label element, insert a selection list for the crust field. Add the following options: Thin, Thick, Hand Tossed, and Deep Dish. Make the option values thin, thick, hand, and dish. Make Hand Tossed the default selection.

**EXPLORE**

7. Insert a label element containing the text "Toppings (select all that apply)". Within the label element, insert the toppings field. Set the size of the selection list to five entries and allow users to make multiple selections.

**EXPLORE**

8. Within the topping selection list, insert the list of toppings from the `topping.txt` file. Break the options into option groups with labels Vegetables and Fruits, Meats, and Cheeses. You do not have to specify option values.
9. Insert a label element with the text Quantity. Give the label the id `qLabel`. Within the label element, insert a selection list for the qty field. Insert the numbers 1 through 10 for both the option text and option values.

**EXPLORE**

10. After the selection list, insert an image control element displaying the image `buttonball.jpg`.
11. Insert a field set with the id `extras` and the legend Extras.
12. Within the field set, insert a label element containing the text "Breadsticks". Inside the label, insert a selection list for the bread field containing the options 6 piece, 12 piece, and 18 piece. The option values are 6, 12, and 18.
13. Repeat Step 12 for the cheesy bread menu item. The label text is "Cheesy Bread" and the field name is `cbread`.
14. Insert a label element containing the text "Chicken Wings", Inside the label, insert a selection list for the wings field containing the options 6 piece, 12 piece, 18 piece, and 24 piece with the values 6, 12, 18, and 24. Make 12 the default selection.
15. Repeat Step 14 for the Barbecue Wings menu item. The label text is "Barbecue Wings" and the field name is `bwings`.
16. After the selection list, insert an image control element displaying the image `buttonball.jpg`.
17. Save your changes to the `pizza.htm` file.
18. Go to the `orderform.css` file in your text editor and add the following styles to the style sheet:
  - Set the margin and padding space of the form element to 0 pixels.
  - Set the bottom margin of the `fieldset` element to 10 pixels. Set the other margin sizes to 0 pixels. Change the background color to ivory. Add a 5-pixel inset border with the color value (255, 192, 192).
  - Display field set legends in a red font with the kerning set to 3 pixels.
  - Display label elements as blocks with relative positioning. Set the width to 400 pixels, set the top margin to 5 pixels, the bottom margin to 10 pixels, and the left/right margins to 0 pixels.
  - Display *select elements* using *absolute* positioning *with* the top coordinate set to *0 pixels and the left coordinate set to 250 pixels*. Set the font size to 12 pixels.
  - For the label with the id `qLabel`, set the top margin to 80 pixels.
  - Display input elements with a left margin of 250 pixels.

19. Save your changes to the file and then open **pizza.htm** in your Web browser. Verify that the design and layout resemble that shown in Figure 6-62.
20. Submit your completed files to your instructor.

## Create

## | Case Problem 4

Test your knowledge of Web forms by creating an order form for an online computer store.

**Data Files needed for this Case Problem: mclogo.jpg**

**Millennium Computers** You are employed at Millennium Computers, a discount mail-order company specializing in computers and computer components. You've been asked by your supervisor, Sandy Walton, to create an order form Web page so that customers can purchase products online. Your order form is for computer purchases only. There are several options for customers to consider when purchasing computers from Millennium. Customers can choose from the following:

- Processor Speed: 2.4 GHz, 3.2 GHz, 4 GHz
- Memory: 1 GB, 2 GB, 4 GB, 8 GB
- Monitor Size: 15", 17", 19", 21"
- Hard Drive: 240 GB, 500 GB, 750 GB, 1 TB
- DVD burner: yes/no
- LAN card: yes/no
- Media card reader: yes/no

Complete the following:

1. Use your text editor to create an HTML file named **pc.htm** and two style sheets named **mill.css** and **oform.css**. Enter **your name** and **the date** in a comment section of each file. Include any other comments you think will aptly document the purpose and content of the files. Save the files in the `tutorial.06\case4` folder included with your Data Files.
2. Design a Web page for the Millennium Computers Web page. Insert any styles you create in the `mill.css` style sheet. You are free to use the **mclogo.jpg** file and whatever text or images you wish to complete the look and content of the Web page.
3. Within the **pc.htm** file, insert a Web form containing the following elements:
  - Input boxes for the customer's first name, last name, street address, city, state, zip code, and phone number. The field names are `fName`, `lName`, `street`, `city`, `state`, `zip`, and `phone`.
  - Selection lists for the processor speed, memory, monitor size, and hard drive size. The field names are `pSpeed`, `mem`, `monitor`, and `hd`. The option values should match the option text.
  - Option buttons for the DVD burner, LAN card, and media card reader options. The field names are `dvd`, `LAN`, and `mCard`.
  - A check box for the warranty field that asks whether customers want the 24-month extended warranty.
  - A text area box requesting additional information or comments on the order.
  - Three form buttons: a submit button with the text "Send Order", a reset button with the text "Cancel Order", and a command button with the text "Contact Me".
  - Name the form `cOrder` and submit the form using the post method to the CGI script located at [http://www.mill\\_computers.com/orders/process.cgi](http://www.mill_computers.com/orders/process.cgi).
4. Create a style for your form in the **oform.css** style sheet. The layout and appearance of the form is up to you.

5. Test your Web site on a variety of browsers to ensure your design works under different conditions.
6. Submit your completed files to your instructor.

## Review

## | Quick Check Answers

**Session 6.1**

1. A CGI script is a program running on a Web server that receives data from a form and uses it to perform a series of tasks.
2. `<form id="registration" name="registration"> ... </form>`
3. 

```
<fieldset id="contactInfo">
    <legend>Contact Information</legend>
</fieldset>
```
4. Either with the `for` attribute (explicitly) or by nesting the control element within the label (implicitly)
5. `clabel for="phone">Phone</label>`
6. `maxlength="10"`
7. `cinput type="text" id="subscribe" name="subscribe" value="Yes" />`
8. `input [type="text"]{display: block}`

**Session 6.2**

1. 

```
<input type="radio" name="Computer" value="PC" />
<input type="radio" name="Computer" value="Macintosh" />
```
2. 

```
cinput type="radio" name="Computer" value="PC"
" checked="checked" />
```
3. 

```
<select id="State" name="State">
    <option>California</option>
    <option>Nevada</option>
    <option>Oregon</option>
    <option>Washington</option>
</select>
```
4. `<option selected="selected">Oregon</option>`
5. `<select id="State" name="State" size="4">`
6. `<select id="State" name="State" multiple="multiple">`
7. 

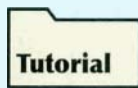
```
clabel for="Computer">I use a PC</label>
cinput type="checkbox" name="Computer" id="Computer" value="Yes" />
```
8. 

```
ctextarea rows="10" cols="40" id="Memo" name="Memo">
</textarea>
```

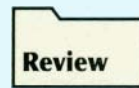
### Session 6.3

1. `<input type="submit" value="Send Form" />`
2. `<input type="reset" text="Cancel Form" />`
3. `<input type="image" id="Sites" name="Sites" src="sites.gif" />`
4. `<input type="hidden" id="Subject" name="Subject" value="Form Responses" />`
5. `<form action="http:www.j_davis.com/cgi-bin/post-query" method="get"> ...</form>`
6. `<form action="mailto:walker@j_davis.com" method="text/plain" />`

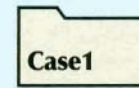
### Ending Data Files



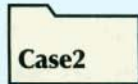
donations.htm  
forms.css  
main.css  
+ 4 graphic files



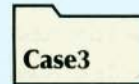
main.css  
vforms.css  
volunteer.htm  
+ 4 graphic files



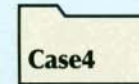
pcg.css  
sform.css  
subscription-htm  
+ 2 graphic files



cw.css  
cwquiz.htm  
qform.css  
+ 2 graphic files



orderform.css  
pizza.htm  
rb.css  
+ 4 graphic files



mill.css  
oform.css  
pc.htm  
+ 1 graphic file