

Web Forms

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

- ◎ Understand how forms work
- ◎ Use the `<form>` element
- ◎ Create input objects
- ◎ Style forms with Cascading Style Sheets (CSS)
- ◎ Build a form

This chapter covers the HTML form elements. Forms let you build interactive Web pages that collect information from a user and process it on the Web server. You can use forms to gather information and create databases or to send customized responses to your users. Forms collect—but do not process—data. The data processing must be performed on the Web server that hosts the form. Forms are the basis for online commerce; without them users would not be able to enter customer address, credit card, and ordering information on the Web.

Understanding How Forms Work

Figure 11-1 shows a typical form. You can use a variety of input elements for the form based on the type of information you want to gather from your user. Forms usually contain basic HTML formatting tags such as `<p>` and `
`. Forms can also be styled with CSS, which helps control their visual layout. Well-designed forms include active white space, aligned form elements, and clear labels. Use the design principles you have learned throughout this book to create forms that are legible and easy to use.

Sample Form - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Sample Form

GoFish! Magazine — Reader Survey

Tell us who you are:

First Name:

Last Name:

Select the species you prefer to fish:

☐ Smallmouth Bass

☐ Largemouth Bass

☐ Crappie

☐ Walleye

☐ Muskie

☐ Pike

If you own a boat, select the type:

Briefly tell us your favorite fish story:

Enter your story here...

Would you like to be on our mailing list?

☐ Yes ☐ No

Figure 11-1 Sample HTML form

The HTML form itself is the interface for the user to enter data, but all of the actual data processing is performed on the server using applications that reside on the Web server or in the Common Gateway Interface (CGI). The traditional method of processing forms input has been to use the **Common Gateway Interface**, which is the communications bridge between the Internet and the server. Using programs called scripts, CGI can collect data sent by a user via the Hypertext Transfer Protocol (HTTP) and transfer it to a variety of data-processing programs, including spreadsheets, databases, or other software running on the server. The data-processing software can work with the data and send a response back to CGI, and then on to the user.

Although CGI is still in use, there are alternatives that are faster and more efficient. Most of the popular Web server software, such as Apache on Linux Web servers and Internet Information Service (IIS) on Windows Web servers, have their own plug-in modules that run forms-processing software.

The programs that manage the processing of the collected data can be written in a variety of programming languages, although scripting languages are the most common. The scripting language most commonly used with HTML forms is **JavaScript**, which was originally created for the Netscape browser in 1995. JavaScript is a client-side scripting language, which means that it runs on the user's computer, rather than on the server. JavaScript can enhance your Web site with beneficial programming functions, including form validation that checks a user's form entries for errors before the result is sent to a form **script**, which is a program that transfers form data to a server. A complete JavaScript tutorial is out of the scope of this book, though many good references are available on the Web



Good sources for JavaScript information on the Web include the following:
www.yourhtmlsource.com/javascript
www.w3schools.com/js/default.asp

A more recent enhancement to forms processing is **AJAX**, which is short for Asynchronous JavaScript and XML. AJAX is a group of technologies that is used on the client to retrieve data from the user and submit the server request in the background. The result is that the data can be processed without the user having to wait for the page to reload, allowing enhanced speed and interactivity. Despite the name AJAX, scripts can be created with languages other than JavaScript and does not need XML to manage the data. You can find out more about scripting and AJAX at www.w3.org/standards/webdesign/script.

Using the <form> Element

The <form> element is the container for creating a form, as the <table> element is the container for the elements that create a table. A form has a number of attributes that describe how the form data is handled, as described in Table 11-1.

Attribute	Description
action	The URL of the application that processes the form data; this URL points to a script file or an e-mail address
enctype	The content type used to submit the form to the server (when the value of the method is “post”); most forms do not need this attribute
method	Specifies the HTTP method used to submit the form data; the default value is “get” <ul style="list-style-type: none">• get—The form data is appended to the URL specified in the action attribute• post—The form data is sent to the server as a separate message
accept	A comma-separated list of content types that a server processing this form can handle correctly; most forms do not need this attribute
accept-charset	A list of allowed character sets for input data that is accepted by the server processing this form; most forms do not need this attribute

Table 11-1 Form Attributes

The <form> element by itself does not create a form. It must contain **form controls** (such as <input> elements) and structural elements such as <p> or to control the look of the form. CSS style classes let you select different form elements and control their look with style rules.

A variety of form controls are available for collection information, as described in the following sections. The following code shows a typical <form> element with some of the attributes listed in Table 11-1. This code specifies that the form data the user enters is being sent to a program named register.asp that resides on a Web server.

```
<form method="post"
action="https://signup.website.com/register.asp">
```

Using get or post

The method you will specify in your form is often defined by the programmers that create the script that process the form data. The difference between *get* and *post* is the way the data is sent to the server.

```
method="get"
```

This method sends the form information by including it in the URL. The data is not secure and should not be used for passwords or other confidential information.

```
method="post"
```

This method sends the form information securely to the server within the message body, so the data is not visible. This is the more common method for sending form data.

Using the mailto Action

With the mailto action, you can collect data from a form and send it to any e-mail address. Although this method does not allow any data processing, it is fine for the occasional brief form or for simple Web sites. The data is sent as a long string of text, which you can make easier to read by including the `enctype="text/plain"` attribute. A form element with a mailto action looks like the following:

```
<form action="mailto:joel@joelsklar.com"
method="post" enctype="text/plain">
```

The data will be sent as an e-mail message to the specified e-mail address. For example, the data from an address form would look like this:

```
name=Joe User
street=3 Maple Lane
city=Smalltown
state=MA
zip=00000
```

The mailto action depends on the user having e-mail client software configured on his or her system. If a mail client is configured, the user's e-mail software will open when the user clicks the submit button. If no mail client is configured, the user will not be able to submit the form data.

Creating Input Objects

The `<input>` element defines many of the form input object types. Table 11-2 lists the available object types. You use the *type* attribute to specify the object type.

Type	Attribute Value	Description
text		Creates a text entry field that lets the user enter a single word or a line of text; this is the default object type
password		Creates the same type of text entry field created by the value "text," but the user entry is masked by asterisks
checkbox		Provides on/off toggles that the user selects; check boxes are best used with multiple-answer questions, and multiple check boxes can contain the same name, letting you group them together so that users can select multiple values for the same property.
radio		Lets a user choose one value from a range of values; when radio buttons are grouped together with the same name, only one choice can be selected
submit		Sends the form data to the server using the transmission method specified in the <form> element; every form needs a submit button
reset		Clears the form of any user-entered data and returns it to its original state
hidden		Adds a control that is not displayed in the browser; the hidden type is useful for sending additional information with the form data that may be needed for processing
image		Adds a graphic button to the form, rather than the default button
button		Creates a button that has no default behavior; the button's function is usually defined by a script; when the user pushes the button, the script function is triggered
file		Lets the user select a file that is submitted with the form

Table 11-2 <input> Element Types



HTML5 offers new form input types that are currently not supported by most browsers. You will find descriptions of these new form input types in Appendix A.

Labeling Form Elements

The <label> element lets you create a caption for an input element. Figure 11-2 shows the label element next to both text and check box elements.

Figure 11-2 The `<label>` element provides captions in a form

The `<label>` element in the following code sample contains the caption text *First Name*:

```
<p>
  <label class="username" >First Name:</label>
  <input type="text" name="firstname"
    size="35" maxlength="35" />
</p>
```

The `<label>` element lets you extend the clickable area of a form element to make them more accessible. For example, rather than having to click only in the check box, users can also click the text caption to select the check box as shown in Figure 11-3.

Figure 11-3 The `<label>` element increases the clickable area

To make the text clickable, you associate the `<label>` element with the `<input>` element by using the *for* and *id* attributes as shown in the following code.

```
<p>
  <label class="username" for="First Name">
    First Name:</label>
  <input type="text" name="firstname" id="First Name"
    size="35" maxlength="35" />
</p>
```

The *for* attribute in the `<label>` element must match the *id* attribute in the `<input>` element to associate the text with the input element.

Creating Text Boxes

The text entry box is the most commonly used `<form>` input type. The default text box is 20 characters long, although you can change this value using the *size* attribute. The user can enter an unlimited number of characters in the text box even though they exceed the visible length. You can constrain the user's entry of text with the *maxlength* attribute and supply a default value for the text with the *value* attribute. The following code shows a simple form with two text boxes.

```
<form action="http://server.com/cgi_bin/script.cgi"
method="post">

<p>Tell us who you are:</p>

<p>
  <label class="username" for="First Name">
    First Name: </label>
  <input type="text" name="firstname" id="First Name"
    size="35" maxlength="35" /></p>
<p>
  <label class="username" for="Last Name">
    Last Name: </label>
  <input type="text" name="lastname" id="Last Name"
    size="35" maxlength="35" /></p>

</form>
```

This code creates the two text box inputs shown in Figure 11-4.

Figure 11-4 Text box input type

Creating Check Boxes

Check boxes are on/off toggles that the user can select. You can use the name attribute to group check boxes, allowing the user to select multiple values for the same property. Figure 11-5 shows an example of a group of check boxes.

Figure 11-5 Check box input type

In the following code, the various fish species check boxes are grouped together with the name attribute set to *species*. Notice that the check boxes reside within a standard unordered list with the CSS list-style-type property set to *none*. Also, the `<label>` element comes after the `<input>` element to display the captions to the right of the check boxes.

```
<ul>
<li>
<input type="checkbox" name="species" value="smbass"
id="Smallmouth Bass" />
<label for="Smallmouth Bass">Smallmouth Bass</label>
</li>

<li>
<input type="checkbox" name="species" value="lgbass"
id="Largemouth Bass" />
<label for="Largemouth Bass">Largemouth Bass</label>
</li>

<li>
<input type="checkbox" name="species" value="crappie"
id="Crappie" />
<label for="Crappie">Crappie</label>
</li>

<li>
<input type="checkbox" name="species" value="walleye"
id="Walleye" />
<label for="Walleye">Walleye</label>
</li>

<li>
<input type="checkbox" name="species" value="muskie"
id="Muskie" />
<label for="Muskie">Muskie</label>
</li>

<li>
<input type="checkbox" name="species" value="pike"
id="Pike" />
<label for="Pike" />Pike</label>
</li>
</ul>
```

To check a check box by default, you can use the `checked` attribute. The following code fragment shows the syntax for this attribute. Here, the Pike check box is checked by default.

```
<input type="checkbox" name="species" value="pike"
checked="checked" />Pike
```

Creating Radio Buttons

Radio buttons are like check boxes, but only one selection is allowed. When radio buttons are grouped with the *name* attribute, only one value can be selected to be “on,” while all other values must be “off.” To preselect one of the radio buttons, you use the `checked` attribute. Figure 11-6 shows the radio buttons input type.



Figure 11-6 Radio buttons input type

In the following code, the *Yes* and *No* radio buttons are grouped together with the name attribute set to *list*. The user can choose only one of the two values. The `<label>` elements provide the captions next to the button.

```
<p>Would you like to be on our mailing list?</p>
<p><input type="radio" name="list" value="yes"
id="Yes" />
<label for="Yes">Yes</label>
<input type="radio" name="list" value="no"
id="No" />
<label for="No">No</label>
</p>
```



Use check boxes when you want to create a question to which multiple answers are allowed. Use radio buttons when you want users to choose only one answer.

Creating Submit and Reset Buttons

The submit and reset button input types let the user either send the form data to be processed or clear the form and start over. These are predefined functions that are activated by the button type. Set the input type to either *submit* or *reset*. The default button text values are *Submit Query* and *Reset*. You can use the *value* attribute to customize the button text. Figure 11-7 shows the buttons.

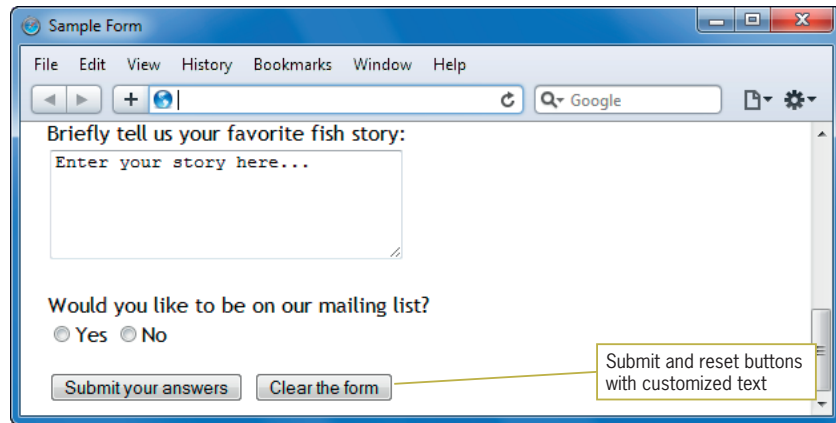


Figure 11-7 Submit and reset input buttons

The following code shows the addition of submit and reset buttons with customized button text shown in the figure.

```
<p>
<input type="submit" value="Submit your answers" />
<input type="reset" value="Clear the form" />
</p>
```

Creating an Image for the Submit Button

You can choose an image file and use it instead of the default button image for the submit button. The image type works only for the submit function. Make sure that the image you choose is an acceptable Web file format (GIF, PNG, or JPG). The src attribute contains the location of the image file. Remember to include an alt attribute as you would with any other image. Figure 11-8 shows the use of an image (submit.jpg) for the submit button.

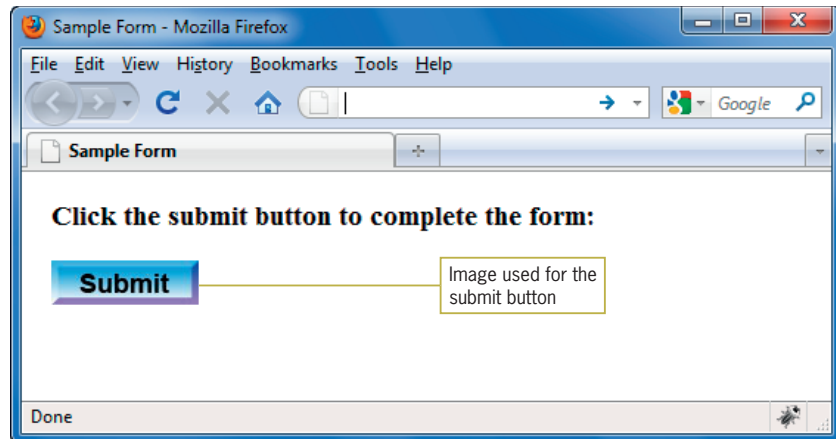


Figure 11-8 Using an image for the submit button

The following code shows the use of an image input type specifying the image and alt attribute.

```
<h3>Click the submit button to complete the form:</h3>
<p><input type="image" src="submit.jpg"
alt="submit button" /></p>
```

Letting the User Submit a File

The file input type object lets users select a file on their own computers and send it to the server. This type lets you create a text input area for the user to enter a filename. The length of the text input is specified with the size attribute. The file type automatically includes a browse button that lets users browse for a file on their computer. This input type looks different in different browsers, as shown in Figure 11-9, which includes both Internet Explorer and Safari results.

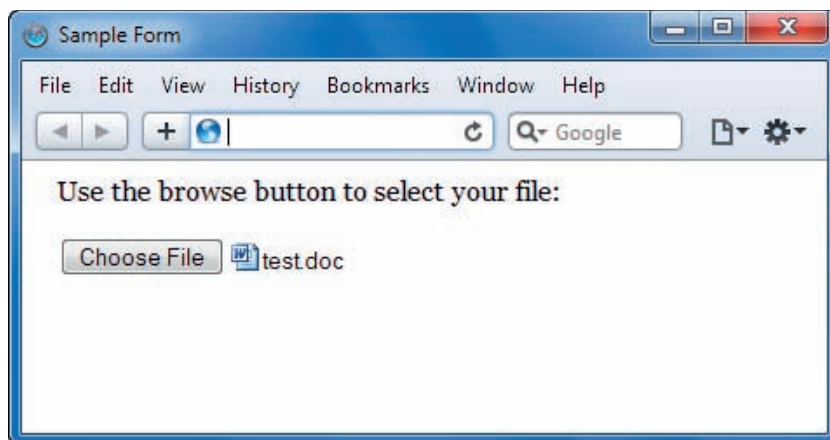
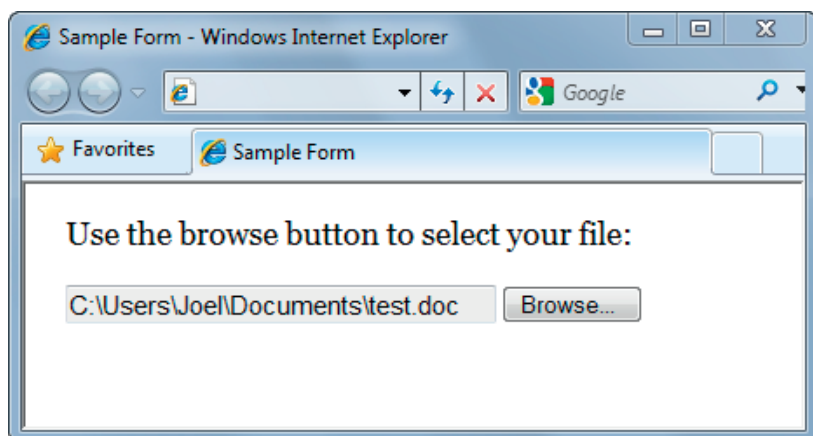


Figure 11-9 File input type in Internet Explorer (top) and Safari (bottom)

The following code shows the file input type. In this case, the text box accepts up to 30 characters.

```
<p>Use the browse button to select your file:</p>
<p><input type="file" size="30" /></p>
```

Creating a Password Entry Field

The password input type object works like a text input box, with the additional feature that the entered text is hidden by asterisks rather than shown on the screen. This is a very low level of password protection, as the password is only protected from unauthorized users looking at the screen. The password itself is sent to the server as plain text, and anyone with network access could read the password information. If you use passwords, check with your system administrator to see whether you can send passwords over a secure Internet connection. Figure 11-10 shows password input type.

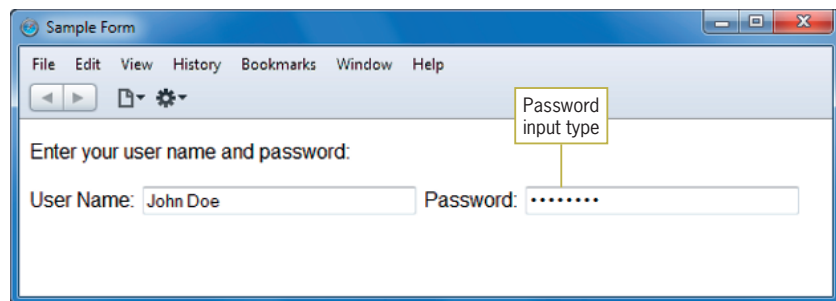


Figure 11-10 Password input type

The following code shows the use of the password input type. Both the user name and password text boxes accept up to 30 characters.

```
<p>Enter your user name and password:</p>
<p>
User Name: <input type="text" size="30" />
Password: <input type="password" size="30" />
</p>
```

Using the <select> Element

The <select> element lets you create a list box or scrollable list of selectable options. The <select> element is a container for the <option> element. Each <option> element contains a list value.

The following code shows the standard type of list box; the user can choose one value from the list. Figure 11-11 shows the result of the code.

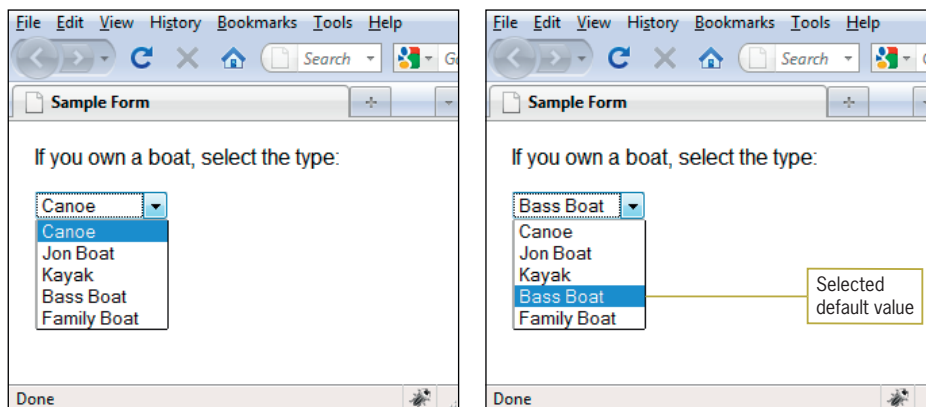


Figure 11-11 Select list element

Notice that the first option in the list is the value that appears in the list box text area.

```
<p>If you own a boat, select the type:</p>
<p>
<select name="boats">
  <option value="canoe">Canoe</option>
  <option value="jonboat">Jon Boat</option>
  <option value="kayak">Kayak</option>
  <option value="bassboat">Bass Boat</option>
  <option value="familyboat">Family Boat</option>
</select>
</p>
```

You can select the default value in a list by adding the *selected* attribute to an `<option>` element. In the following list, “Bass Boat” is the default value, as shown in the browser window on the right in Figure 11-11.

```
<p>If you own a boat, select the type:</p>
<p>
<select name="boats">
  <option value="canoe">Canoe</option>
  <option value="jonboat">Jon Boat</option>
  <option value="kayak">Kayak</option>
  <option value="bassboat" selected="selected">
    Bass Boat</option>
  <option value="familyboat">Family Boat</option>
</select>
</p>
```

You can also let the user pick multiple values from the list by adding the *multiple* attribute to the `<select>` element. The user can hold the Ctrl key and select multiple items in the list, or hold the Shift key to select contiguous options. Figure 11-12 and the following code show the use of the *multiple* attribute.

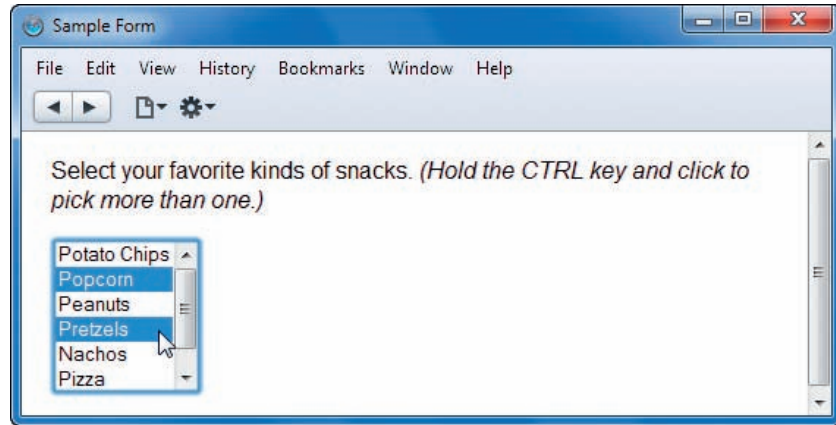


Figure 11-12 Scrollable select list with multiple choices

The size attribute specifies how many list options are visible at a time. The following list shows six options at once.

```
<p>Select your favorite kinds of snacks. <em>(Hold the CTRL key and click to pick more than one.)</em></p>
<p>
<select name="snacks" multiple size="6">
  <option>Potato Chips</option>
  <option>Popcorn</option>
  <option>Peanuts</option>
  <option>Pretzels</option>
  <option>Nachos</option>
  <option>Pizza</option>
  <option>Fries</option>
</select>
</p>
```

Grouping List Options

You can group and label sets of list options with the `<optgroup>` element and label attribute. The result is a heading for a series of options within a list. Figure 11-13 shows the result of using the `<optgroup>` element.

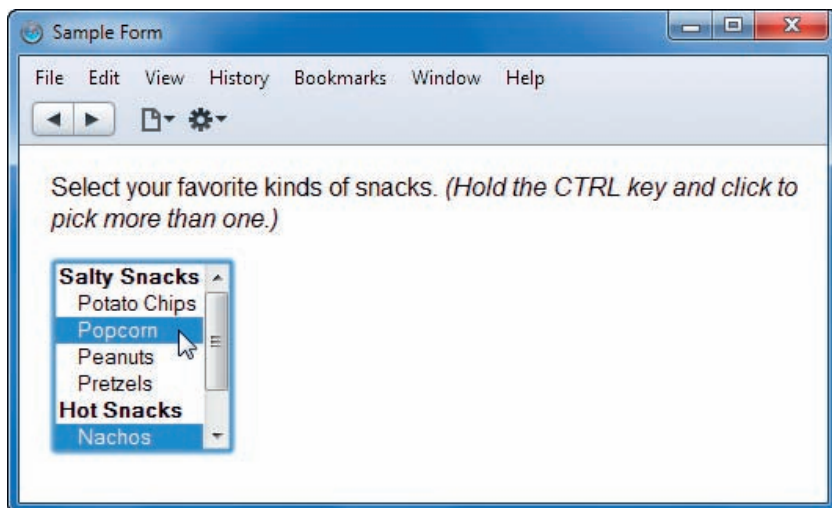


Figure 11-13 Grouping list options

The browser determines the format of the labels, but they are usually displayed in bold text. In the following code, the snack list is divided into two groups: salty snacks and hot snacks.

```
<p>
<select name="snacks" multiple size="7">
  <optgroup label="Salty Snacks">
    <option>Potato Chips</option>
    <option>Popcorn</option>
    <option>Peanuts</option>
    <option>Pretzels</option>
  </optgroup>
  <optgroup label="Hot Snacks">
    <option>Nachos</option>
    <option>Pizza</option>
    <option>Fries</option>
  </optgroup>
</select>
</p>
```

Using the <textarea> Element

The <textarea> element lets you create a text area for user input larger than the <input> text type object described previously. Figure 11-14 shows a sample of a <textarea> element.

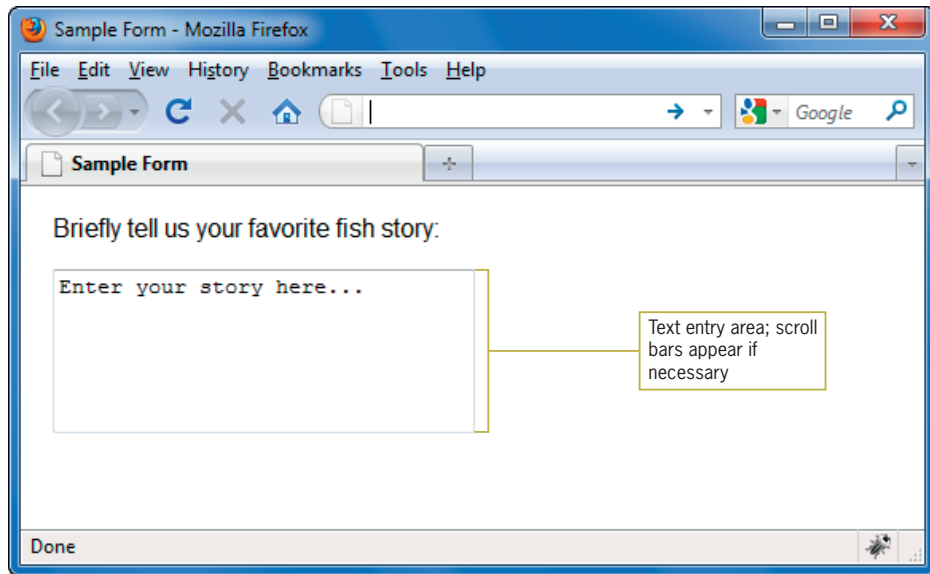


Figure 11-14 Using the `<textarea>` element

You can specify the width and height of the text area with the `cols` and `rows` attributes. If the text entered is longer than the height allotted, the browser adds scroll bars automatically. Any text you enter in the `<textarea>` element appears as the default text in the user's browser. The following code shows a text area set to 30 columns wide by 5 rows high.

```
<p>Briefly tell us your favorite fish story:</p>
<p>
<textarea name="fishstory" rows="5" cols="30">
Enter your story here...
</textarea>
</p>
```

Creating Input Groupings

You can use the `<fieldset>` and `<legend>` elements to create groupings of different types of `<input>` elements. The `<fieldset>` element contains the `<input>` elements, and the `<legend>` element contains a label for the grouping. These two elements help make your forms more readable and increase their accessibility to screen readers and other accessibility devices. Figure 11-15 shows the use of the `<fieldset>` and `<legend>` elements. The code for the page follows.

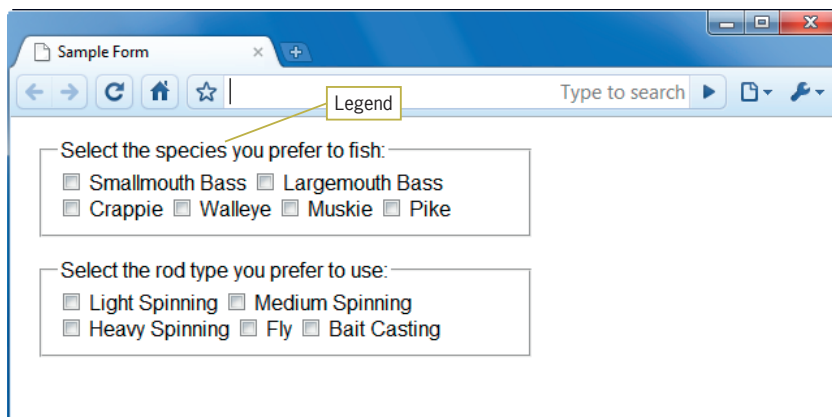


Figure 11-15 Grouping and labeling <input> elements

```

<fieldset>
<legend>Select the species you prefer to fish:
</legend>
<p>
<input type="checkbox" name="species"
value="smbass" />
Smallmouth Bass
<input type="checkbox" name="species"
value="lgbass" />
Largemouth Bass <br/>
<input type="checkbox" name="species"
value="crappie" />
Crappie
<input type="checkbox" name="species"
value="walleye" />
Walleye
<input type="checkbox" name="species"
value="muskie" />
Muskie
<input type="checkbox" name="species"
value="pike" />
Pike
</p>
</fieldset>

<fieldset>
<legend>Select the rod type you prefer to use:
</legend>
<p>
<input type="checkbox" name="species"
value="ltspin" />
Light Spinning
<input type="checkbox" name="species"
value="mdspin" />
Medium Spinning <br/>
<input type="checkbox" name="species"

```

```
value="hvspin" />  
Heavy Spinning  
<input type="checkbox" name="species"  
value="fly" />  
Fly  
<input type="checkbox" name="species"  
value="btcas" />  
Bait Casting  
</p>  
</fieldset>
```

Styling Forms with CSS

Most forms can benefit from CSS styling to increase their legibility and appeal. As you have seen in the form samples in this chapter, forms need at least basic formatting elements, such as `
` and `<p>`, to place form elements on separate lines and add white space. Even with these basic formatting elements, the look of your form may not be acceptable. Figure 11-16 shows a typical form. Notice how the left justification of the form elements gives a ragged look to the form.

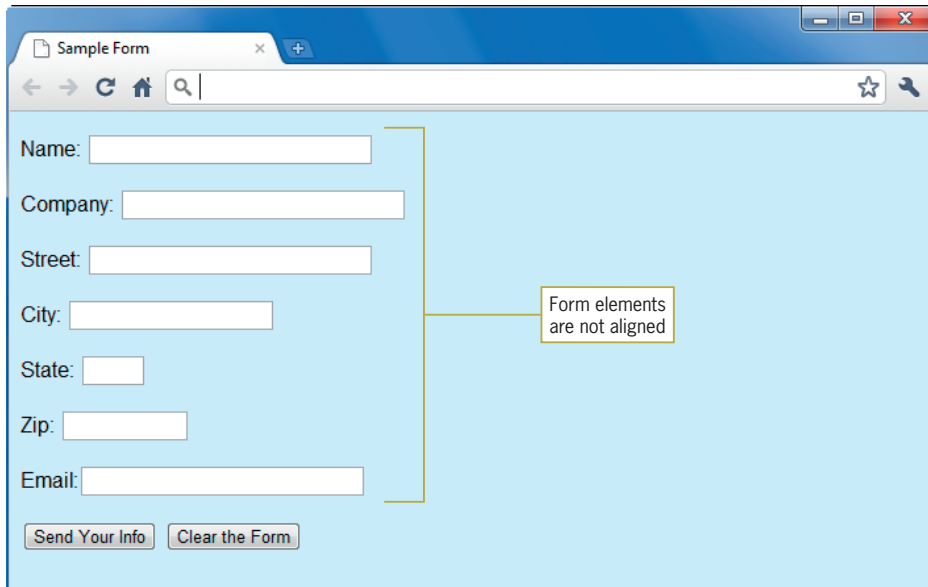


Figure 11-16 Typical form layout

In contrast to Figure 11-16, the form in Figure 11-17 has been styled with CSS, producing a more visually appealing form that is easier for the user to follow when entering data.

The screenshot shows a web browser window with a form titled "Sample Form". The form contains the following elements:

- Name:
- Company:
- Street:
- City:
- State:
- Zip:
- Email:
- Buttons: "Send Your Info" and "Clear the Form"

Figure 11-17 Form layout enhanced with CSS

Adding CSS to form elements is no different than adding styles to standard content elements. You can use many of the CSS properties to specify type styles, background colors, box properties, and colors to enhance the look of your forms. The grouping and labeling elements `<fieldset>`, `<legend>`, and `<label>` are useful when applying styles to forms. You can also use classes or ids to apply styles to specific form elements.

Aligning Form Elements

As shown in Figure 11-17, aligned form elements create a more organized and easy-to-use form. To create this alignment, you can set a width for the labels so they align in a column to the left of the form input elements. First, examine the code that creates the form in Figure 11-17.

```
<form method="post" action="http://someserver/
cgi_bin/script.cgi">
<p>
<label for="name">Name:</label>
<input type="text" size=30 maxlength=256 name="Name"
id="name" />
</p>
```

```
<p>
<label for="company">Company:</label>
<input type="text" size=30 maxlength=256
name="company" id="company" />
</p>

<p>
<label for="street">Street:</label>
<input type="text" size=30 maxlength=256
name="street" id="street" />
</p>

<p><label for="city">City:</label>
<input type="text" size=20 maxlength=256
name="city" id="city" />
</p>

<p>
<label for="state">State:</label>
<input type="text" size=2 maxlength=256
name="state" id="state" />
</p>

<p>
<label for="zip">Zip:</label>
<input type="text" size=10 maxlength=256
name="zip" id="zip" />
</p>

<p>
<label for="email">Email:</label>
<input type="text" size=30 maxlength=256
name="email" id="email" />
</p>

<p>
<input class="submit" type="submit"
value="Send Your Info" />
<input type="reset" value="Clear the Form" />
</p>
</form>
```

Notice that the label and each corresponding form element are contained within a <p> element. You can float the label to the left of the form element within each paragraph. Use the label element as the selector for the style rule. Float the label to the left and set a width that fits the label content as shown:

```
label {
  float: left;
  width: 6em;
}
```

Figure 11-18 shows the result of this style rule. In this figure, a border shows the width and placement of the `<label>` element.

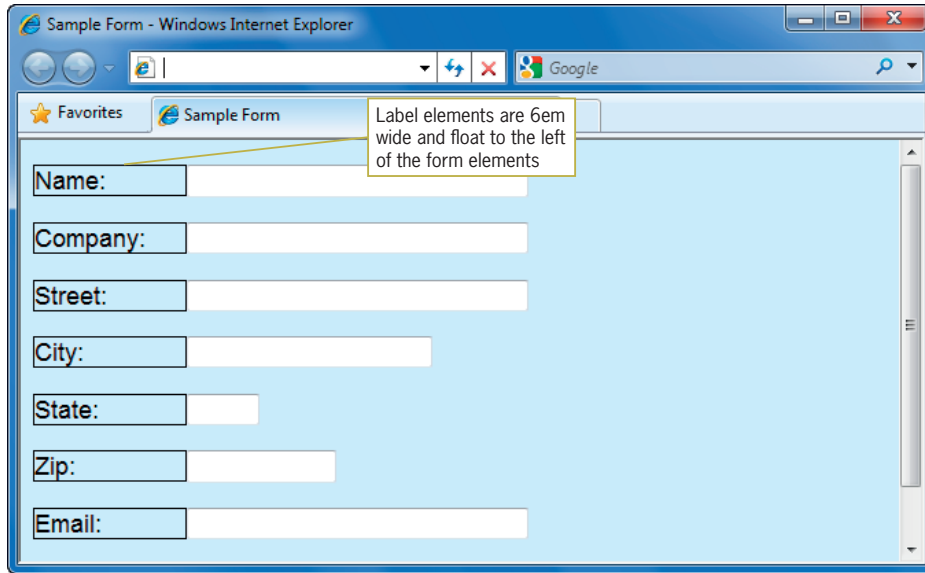


Figure 11-18 Label elements with width and float properties applied

To complete the labels, add style properties to right-align the text and add a right margin to separate the label from the form input element.

```
label {
    float: left;
    width: 6em;
    text-align: right;
    margin-right: 10px;
}
```

Create a style for the paragraph that contains the submit and reset buttons. This rule uses *submit* as the class name and offsets the buttons from the left margin by 8ems.

```
.submit {margin-left: 8em;}
```

Then apply this rule to the paragraph that contains the submit and reset buttons.

```
<p class="submit" >
<input type="SUBMIT" value="Send Your Info" />
<input type="RESET" value="Clear the Form" />
</p>
```

These style rules result in the finished form shown in Figure 11-19.

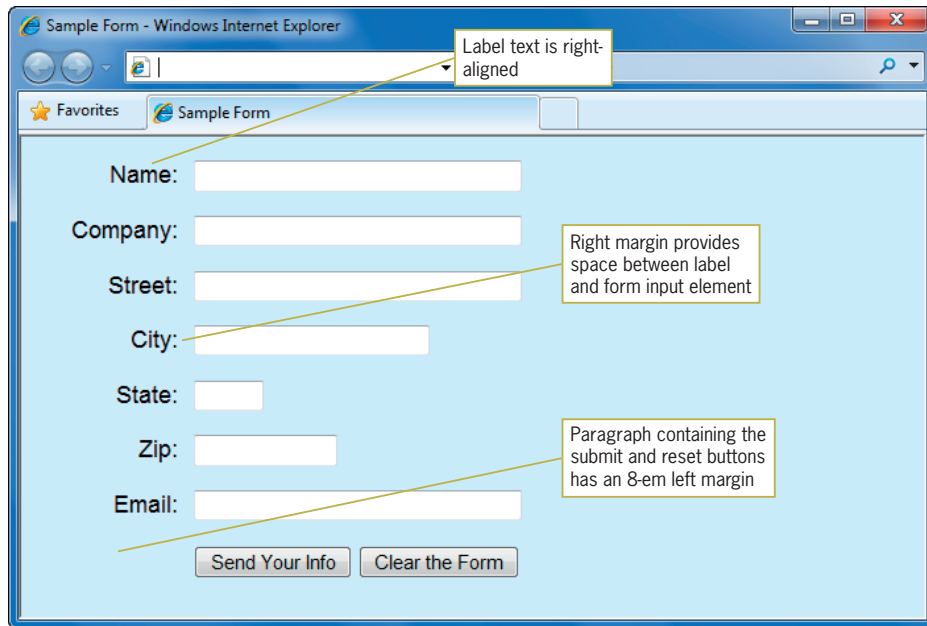


Figure 11-19 Form layout enhanced with CSS



Using an em measurement value for the width makes sure that the label boxes increase in size if the user changes their screen font size.

Styling Fieldset and Legend Elements

The `<fieldset>` and `<legend>` elements are great for applying styles to make your forms more attractive. Figure 11-20 shows the form with `<fieldset>` and `<legend>` elements with their default display behavior. Notice that the fieldset border extends to the width of the browser window. Also, this form already has the labels and form input elements aligned as you saw in the previous section.

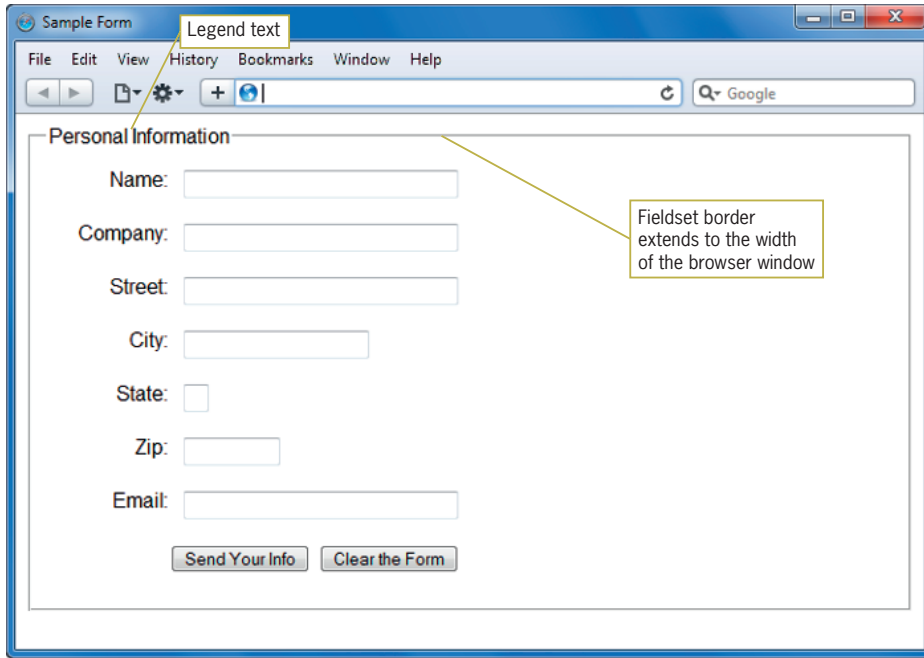


Figure 11-20 Fieldset and legend elements default display

The fieldset can be set to a measured width to keep it from extending to the width of the browser window. The following rule selects the `<fieldset>` element and applies a width of 24em.

```
fieldset {
    width: 24em;
}
```

Adding a background color (`#ddeeff`) and a dark blue border (`#053972`) will help the form stand out.

```
fieldset {
    width: 24em;
    background-color: #ddeeff;
    border: solid medium #053972;
}
```

The legend text can be styled to make it stand out. In this example, the legend had a 1px border that matches the color of the fieldset border. A white background and 2 pixels of padding help offset the text from the fieldset border.

```
legend {  
    border: solid 1px #053972;  
    background-color: #fff;  
    padding: 2px;  
}
```

Figure 11-21 shows the results of the fieldset and legend style rules.

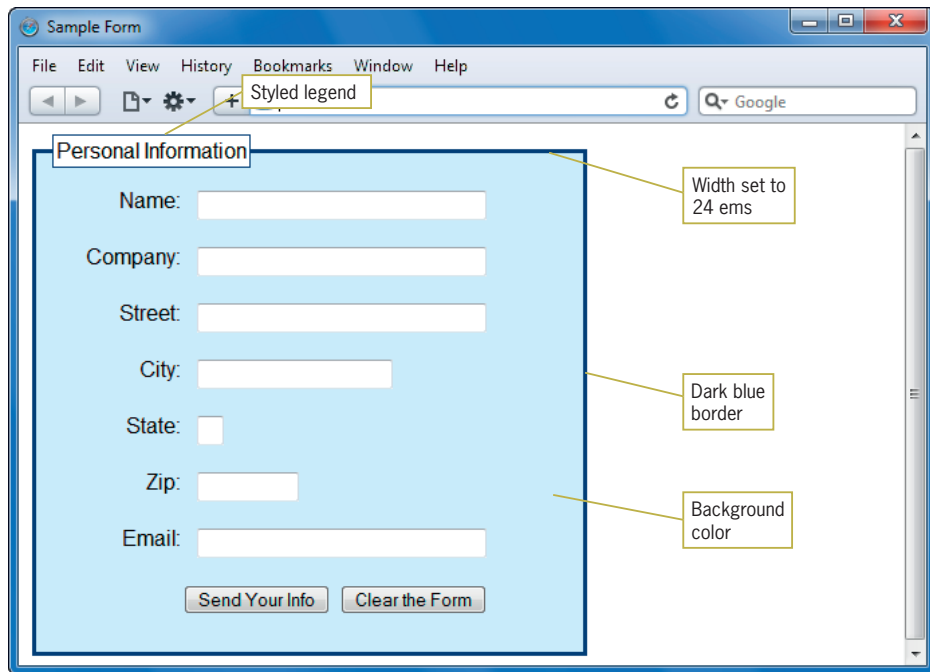


Figure 11-21 Using fieldset and legend style rules

Adding a Background Image

You can add a background image to a fieldset with the `background-image` property as shown in Figure 11-22.

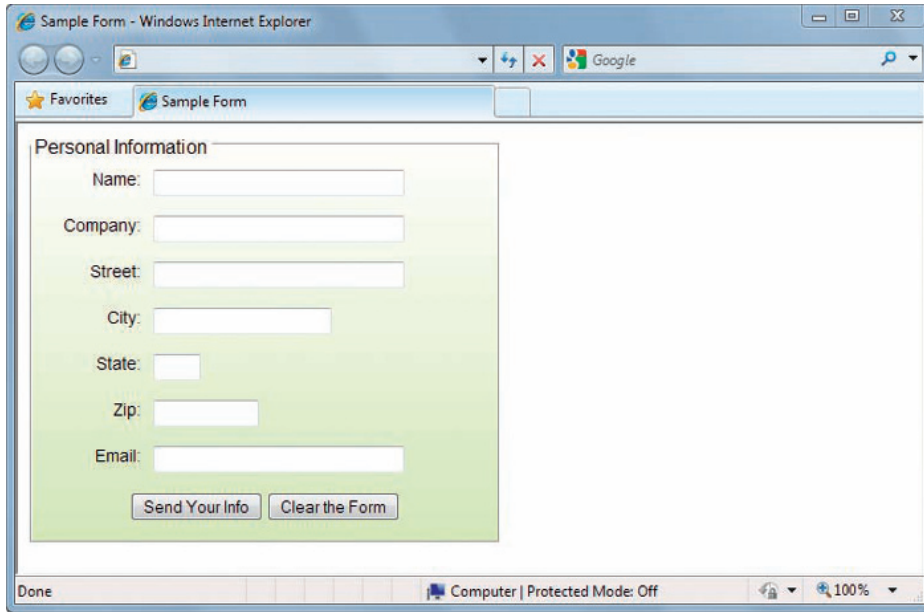


Figure 11-22 Gradient background in a form

In this example, the gradient image (formgradient.jpg) is specified with the background-image property. The background-position is set to bottom, and the background-repeat is set to repeat the graphic along the horizontal axis of the form.

```
fieldset {
    width: 24em;
    background-image: url(formgradient.jpg);
    background-position: bottom;
    background-repeat: repeat-x;
}
```

Activity: Building a Form

In the following set of steps, you will build a form for an online job search service. Users of the service will enter address and personal information into the form.

To begin building the form:

1. Copy the file **form.html** from the Chapter11 folder provided with your Data Files to the Chapter11 folder in your work folder. (Create the Chapter11 work folder, if necessary.)
2. Open **form.html** in your HTML editor, save it as **wonderform.html**, and then examine the code. Note

that a style section contains a body style that sets the font family and left margin for the page.

```
<!DOCTYPE HTML>

<html>
<head>
<title>Wonder Software Online Job Search Form</title>
<meta content="text/html; charset=utf-8"
      http-equiv="Content-Type" />
<style type="text/css">
body {font-family: arial, sans-serif; margin-left: 20px;}
</style>
</head>
<body>
<h1>Wonder Software<br/>Online Job Search</h1>
<form action=" " method="post">
</form>
</body>
</html>
```

3. Begin building the form by adding three text `<input>` elements, one each for the user's name, e-mail address, and telephone number. Set the *size* and *name* attribute values as shown in the following code. Make sure each input element is contained within a `<p>` element.

```
<p>Name: <input size="30" name="name" id="name"></p>
<p>Email: <input size="30" name="email" id="email"></p>
<p>Phone: <input size="30" name="phone" id="phone"></p>
```

4. Add label elements for each text box's label. Make sure to associate the label with the form element by adding the *for* attribute that matches the form input's *id* value.

```
<p><label for="name">Name:</label><input size="30"
  name="name" id="name"></p>
<p><label for="email">Email:</label><input size="30"
  name="email" id="email"></p>
<p><label for="phone">Phone:</label><input size="30"
  name="phone" id="phone"></p>
```

5. Group this set of fields with a `<fieldset>` and accompanying `<legend>` element, as shown in the following code.

```
<fieldset>
<legend>Contact Information</legend>
<p><label for="name">Name:</label><input size="30"
  name="name" id="name"></p>
<p><label for="email">Email:</label><input size="30"
  name="email" id="email"></p>
<p><label for="phone">Phone:</label><input size="30"
  name="phone" id="phone"></p>
</fieldset>
```

6. Save **wonderform.html** and leave it open for the next set of steps. Then view the file in the browser; it should now look like Figure 11-23.

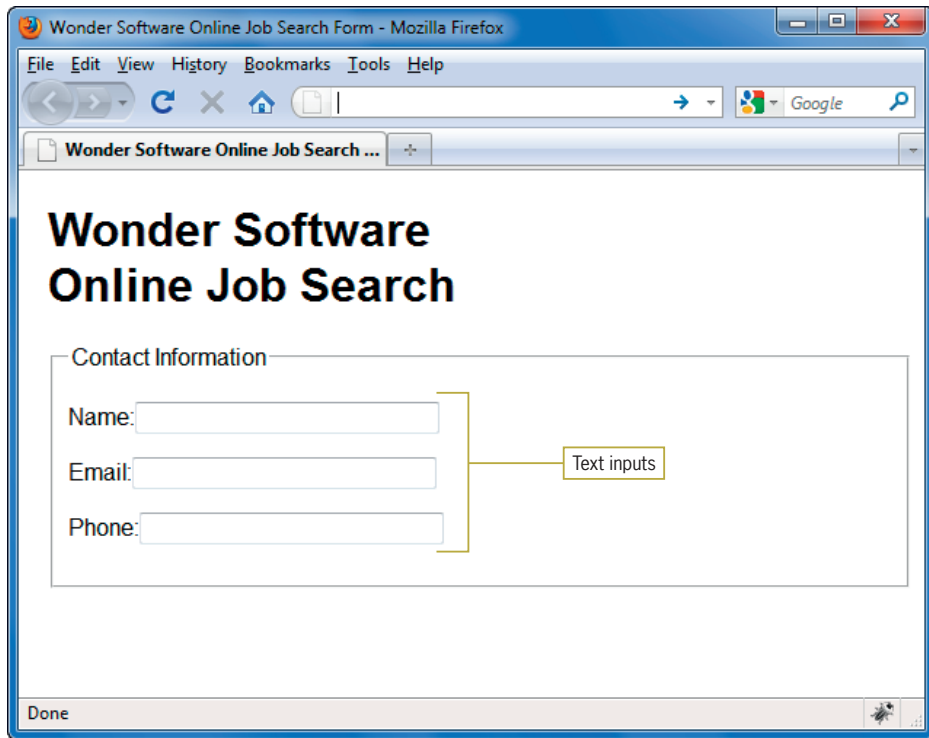


Figure 11-23 Form with three text `<input>` elements

Adding Check Boxes

Continue to build the form by adding check box `<input>` elements to collect information from the user. You will also add a list box of job title types, where users can make only one selection from the list.

To continue building the form:

1. Continue working in the file **wonderform.html**.
2. Add text to identify the check box inputs.

Select Your Area(s) of Interest:

3. Add the check box inputs as shown. The *name* attribute groups the check boxes together under the category *jobtitle*. Each check box is identified with a unique value.

The check boxes are contained in two `<p>` elements, the first with two job titles and the second with three.

```
<p>
<input type="checkbox" name="jobtitle" value="ae" />
Account Executive
```

```
<input type="checkbox" name="jobtitle" value="bd" />
Business Development
</p>
```

```
<p>
<input type="checkbox" name="jobtitle" value="is" />
Inside Sales
```

```
<input type="checkbox" name="jobtitle" value="sm" />
Sales Manager
```

```
<input type="checkbox" name="jobtitle" value="vp" />
VP Sales
</p>
```

4. Add label elements for each check box.

```
<p><input type="checkbox" name="jobtitle" value="ae" />
<label>Account Executive</label>
```

```
<input type="checkbox" name="jobtitle" value="bd" />
<label>Business Development</label></p>
```

```
<p><input type="checkbox" name="jobtitle" value="is" />
<label>Inside Sales</label>
```

```
<input type="checkbox" name="jobtitle" value="sm" />
<label>Sales Manager</label>
```

```
<input type="checkbox" name="jobtitle" value="vp" />
<label>VP Sales</label></p>
```

5. Associate the labels with each checkbox input element by adding *for* and *id* attributes to the `<label>` and `<input>` elements.

```
<p><input type="checkbox" name="jobtitle" value="ae"
id="ae" />
<label for="ae">Account Executive</label>
```

```

<input type="checkbox" name="jobtitle" value="bd"
id="bd" />
<label for="bd">Business Development</label></p>

<p><input type="checkbox" name="jobtitle" value="is"
id="is" />
<label for="is">Inside Sales</label>

<input type="checkbox" name="jobtitle" value="sm"
id="sm" />
<label for="sm">Sales Manager</label>

<input type="checkbox" name="jobtitle" value="vp"
id="vp" />
<label for="vp">VP Sales</label></p>

```

6. Group this set of fields with a `<fieldset>` and accompanying `<legend>` element, as shown in the following code.

```

<fieldset>
<legend>Select Your Area(s) of Interest:</legend>

<p><input type="checkbox" name="jobtitle" value="ae"
id="ae" /><label for="ae">Account Executive</label>

<input type="checkbox" name="jobtitle" value="bd" id="bd" />
<label for="bd">Business Development</label></p>

<p><input type="checkbox" name="jobtitle" value="is"
id="is" /><label for="is">Inside Sales</label>

<input type="checkbox" name="jobtitle" value="sm"
id="sm" /><label for="sm">Sales Manager</label>

<input type="checkbox" name="jobtitle" value="vp"
id="vp"> <label for="vp">VP Sales</label></p>

</fieldset>

```

7. Save **wonderform.html** and leave it open for the next set of steps. Then view the file in the browser; it should now look like Figure 11-24.

Figure 11-24 Form with check box `<input>` elements

Adding a List Box and Radio Buttons

Continue to build the form by adding two more `<input>` elements to collect information from the user. You will add a list box of job position options and a question with a yes or no answer.

To continue building the form:

1. Continue working in the file **wonderform.html**.
2. Add a `<select>` element with four blank `<option>` tags, as shown in the following code. Place this code after the closing `<fieldset>` tag from the previous procedure.

```
<p>
Select the type of position you desire:
<select name="position">
<option> </option>
<option> </option>
<option> </option>
```



```
<option> </option>
</select>
</p>
```

3. Fill in a value for each option.

```
<p>
Select the type of position you desire:
<select name="position">
<option>Part-time contract</option>
<option>Full-time contract</option>
<option>Part-time permanent</option>
<option>Full-time permanent</option>
</select>
</p>
```

4. Below the select list, add the <p> element with the following question:

```
<p>
Are you willing to relocate?
</p>
```

5. Add two <input> elements with the type set to “radio” to create radio buttons.

```
<p>
Are you willing to relocate?
Yes <input type="radio" />
No <input type="radio" />
</p>
```

6. Add a value attribute for each element. Set the value for the Yes button to *yes*. Set the No button to *no*. Also, add a name attribute that groups the radio buttons together with a value of *relocate*.

```
<p>
Are you willing to relocate?
Yes <input type="radio" value="yes" name="relocate" />
No <input type="radio" value="no" name="relocate" />
</p>
```

7. Add labels with *for* and *id* attributes.

```
<p>
Are you willing to relocate?
<label for="yes">Yes</label><input type="radio"
value="yes" name="relocate" id="yes" />
<label for="no">No</label><input type="radio"
value="no" name="relocate" id="no"/>
</p>
```

8. Save **wonderform.html** and leave it open for the next set of steps. Then view the file in the browser; it should now look like Figure 11-25.

Wonder Software Online Job Search

Contact Information

Name:

Email:

Phone:

Select Your Area(s) of Interest:

☐ Account Executive ☐ Business Development

☐ Inside Sales ☐ Sales Manager ☐ VP Sales

Select the type of position you desire: Select list input

Are you willing to relocate?

Yes ☐ No ☐ Radio button inputs

Figure 11-25 Adding a select list and radio buttons

Adding Submit and Reset Buttons

You finish the form by adding the submit and reset buttons.

To continue building the form:

1. Continue working in the file **wonderform.html**.
2. Add submit and reset button element types, and set values for each button as shown.

```
<p>
<input type="submit" value="Submit" />
<input type="reset" value="Clear" />
</p>
```

3. Save **wonderform.html** and leave it open for the next set of steps. Then view the file in the browser; it should now look like Figure 11-26.

Wonder Software Online Job Search Form - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Wonder Software Online Job Search ...

Wonder Software Online Job Search

Contact Information

Name:

Email:

Phone:

Select Your Area(s) of Interest:

☐ Account Executive ☐ Business Development

☐ Inside Sales ☐ Sales Manager ☐ VP Sales

Select the type of position you desire:

Are you willing to relocate?

Yes ☐ No ☐

Submit and reset buttons

Done

Figure 11-26 Submit and reset buttons

Styling the Labels

You will create two label styles, one for the Contact Information labels and one for the Areas of Interest labels.

To style the labels:

1. Continue working in the file **wonderform.html**.
2. Create a style for contact labels using the class name *label*. Set the properties as shown to create floating labels with right-aligned text and a margin to offset the label from the input element.

```
label.contact {
    width: 5em;
    float: left;
    text-align: right;
    margin-right: 10px;
}
```

3. Apply the style by adding the class attribute to the contact labels.

```
<fieldset>
<legend>Contact Information</legend>
<p><label for="name" class="contact">Name:</label>
    <input size="30" name="name" id="name"></p>
<p><label for="email" class="contact">Email:</label>
    <input size="30" name="email" id="email"></p>
<p><label for="phone" class="contact">Phone:</label>
    <input size="30" name="phone" id="phone"></p>
</fieldset>
```

4. Create a style for the area labels using the class name *area*. Set the properties as shown to add left and right margins to adjust the spacing of the labels for legibility.

```
label.area {
    margin-left: 2px;
    margin-right: 10px;
}
```

5. Apply the style by adding the class attribute to the area labels.

```
<fieldset>
<legend>Select Your Area(s) of Interest:</legend>

<p><input type="checkbox" name="jobtitle" value="ae"
    id="ae" /><label for="ae" class="area">Account
    Executive</label>

<input type="checkbox" name="jobtitle" value="bd"
    id="bd" /><label for="bd" class="area">Business
    Development</label></p>

<p><input type="checkbox" name="jobtitle" value="is"
    id="is" /><label for="is" class="area">Inside
    Sales</label>

<input type="checkbox" name="jobtitle" value="sm" id="sm" />
    <label for="sm" class="area">Sales Manager</label>

<input type="checkbox" name="jobtitle" value="vp" id="vp">
    <label for="vp" class="area">VP Sales</label></p>
</fieldset>
```

6. Save **wonderform.html** and leave it open for the next set of steps. Then view the file in the browser; it should now look like Figure 11-27.

Wonder Software Online Job Search Form - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Wonder Software Online Job Search ...

Wonder Software Online Job Search

Contact Information

Name:

Email:

Phone:

Select Your Area(s) of Interest:

☐ Account Executive ☐ Business Development

☐ Inside Sales ☐ Sales Manager ☐ VP Sales

Select the type of position you desire:

Are you willing to relocate?

Yes ☐ No ☐

Done

Figure 11-27 Label styles

Styling the Fieldsets and Legends

Finish the form design by styling the fieldsets and the legends.

To style the fieldsets and the legends:

1. Continue working in the file **wonderform.html**.
2. Create a style that selects both fieldsets. Set the width to 24em.

```
fieldset {
    width: 24em;
}
```

3. Provide some white space between the two fieldsets by writing a rule that selects only the area fieldset and applying a top margin of 20px. Use a class name *area* for this rule.

```
fieldset.area {
    margin-top: 20px;
}
```

4. Apply this rule to the area fieldset as shown.

```
<fieldset class="area">
<legend>Select Your Area(s) of Interest:</legend>
<p><input type="checkbox" name="jobtitle" value="ae"
    id="ae" /><label for="ae">Account Executive</label>
<input type="checkbox" name="jobtitle" value="bd" id="bd" />
    <label for="bd">Business Development</label></p>
<p><input type="checkbox" name="jobtitle" value="is"
    id="is" /><label for="is">Inside Sales</label>
<input type="checkbox" name="jobtitle" value="sm"
    id="sm" /><label for="sm">Sales Manager</label>
<input type="checkbox" name="jobtitle" value="vp"
    id="vp"> <label for="vp">VP Sales</label></p>
</fieldset>
```

5. Create a style for the legends that will apply to the <legend> element in both fieldsets.

```
legend {
    font-weight: bold;
}
```

6. Save **wonderform.html**. Then view the file in the browser; it should now look like Figure 11-28.

Wonder Software Online Job Search

Contact Information

Name:

Email:

Phone:

Select Your Area(s) of Interest:

☐ Account Executive ☐ Business Development

☐ Inside Sales ☐ Sales Manager ☐ VP Sales

Select the type of position you desire:

Are you willing to relocate? Yes ☐ No ☒

Figure 11-28 Completed form

Chapter Summary

A usable form interface is the result of choosing the correct form elements for the type of data you are requesting and designing a clear and readable form. Keep the following points in mind:

- You need to work with some type of server-based software program to process the data from your form. An HTML form is the interface for the user to enter data, and the data processing is performed on the server using applications called scripts that usually reside in the Common Gateway Interface (CGI).

- The `<form>` element is the container for creating a form. A form has a number of attributes that describe how the form data is handled, such as *action*, which often specifies the URL of a script file to process the form data.
- You have a variety of form elements to choose from when building a form. The `<input>` element defines many of the form input object types. Use the correct type of input object for the type of data you are gathering. For example, use check boxes for multiple-choice questions. For a long list of choices, use a select list.
- The `<fieldset>` and `<legend>` elements let you create more visually appealing forms that have logical groupings of `<input>` elements with a title.
- Most forms should be formatted to improve their legibility. The most basic formatting elements are `
` and `<p>`, which place form elements on separate lines and add white space. You can avoid the ragged look of forms by using CSS to align form elements, style labels, legends, and fieldsets, or add background colors or graphics.

Key Terms

AJAX—A group of technologies that is used on the client to retrieve data from the user and submit the form request in the background.

Common Gateway Interface (CGI)—A communications bridge between the Internet and the server used as the traditional method of processing forms input. Using programs called scripts, CGI can collect data sent by a user via the Hypertext Transfer Protocol (HTTP) and transfer it to a variety of data-processing programs including spreadsheets, databases, or software running on the server.

form control—An input element in an HTML form, such as a radio button, text box, or check box.

JavaScript—A client-side scripting language used with HTML forms.

script—A program that transfers form data to a server.

Review Questions

1. What are the five commonly supported form elements?
2. What does the action attribute in the <form> element contain?
3. What are the two possible values of the form method attribute?
4. How can you group multiple check boxes together?
5. How are radio buttons different from check boxes?
6. How do you control the length of a user's entry in a text <input> element?
7. How do you enter default text in a text <input> element?
8. How do you force a check box to be selected by default?
9. What button must be included with every form?
10. How do you change the default button image for the submit button?
11. What input type lets the user attach a file to the form data?
12. What is the security problem with the password input type?
13. What are the two types of select lists?
14. What attributes let you specify the width and height of the <textarea> element?

Hands-On Projects

1. In this project, you will build text box form elements.
 - a. In your HTML editor, open the file **blankform.html** in the Chapter11 folder in your work folder.
 - b. Save the file as **textbox.html** in the same location.

- c. Examine the code. The file contains only the default HTML elements and an empty <form> element.
- d. Build the form shown in Figure 11-29. Refer to the following table for each form element’s attribute values.

Name	Size	Maxlength
Street	20	35
City	20	35
State	2	35
Zip	10	35

Table 11-3 Attribute Values for the Text Box Form

- e. Use CSS style properties to align the labels as shown in Figure 11-29.

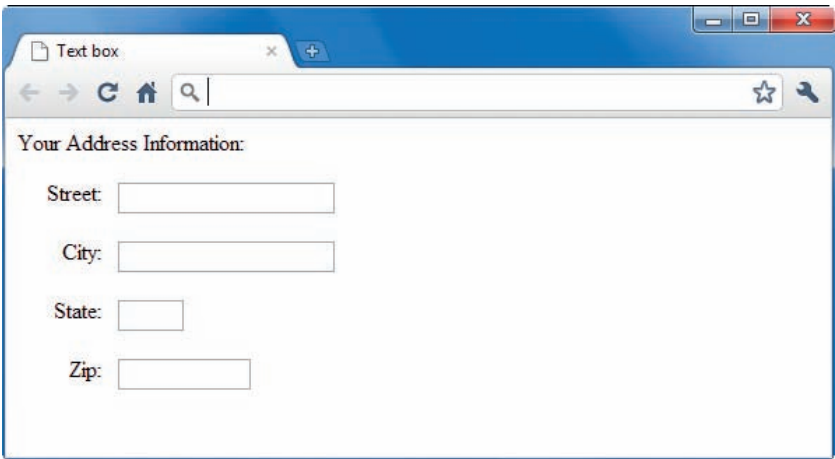


Figure 11-29 Form with text boxes

- 2. In this project, you will build check box form elements.
 - a. In your HTML editor, open the file **blankform.html** in the Chapter11 folder in your work folder.
 - b. Save the file as **checkboxbox.html** in the same location.
 - c. Examine the code. The file contains only the default HTML elements and an empty <form> element.
 - d. Build the form shown in Figure 11-30.

- e. Group the check boxes with a name attribute set to “flavor.”
- f. Add labels that are associated with each input check box.

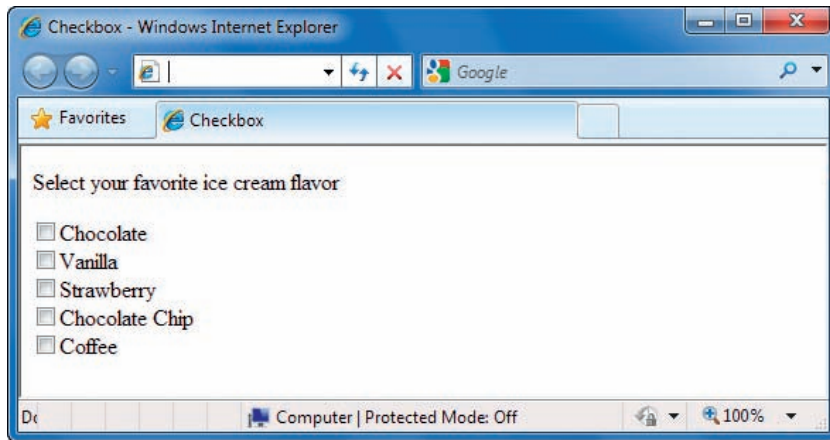


Figure 11-30 Form with check boxes

3. In this project, you build radio button form elements.
 - a. In your HTML editor, open the file **blankform.html** in the Chapter11 folder in your work folder.
 - b. Save the file as **radio.html** in the same location.
 - c. Examine the code. The file contains only the default HTML elements and an empty `<form>` element.
 - d. Build the form shown in Figure 11-31.
 - e. Make sure that “Yes” is the selected option.
 - f. Group the radio buttons with a name attribute set to “offer.”
 - g. Add labels that are associated with each input radio button.

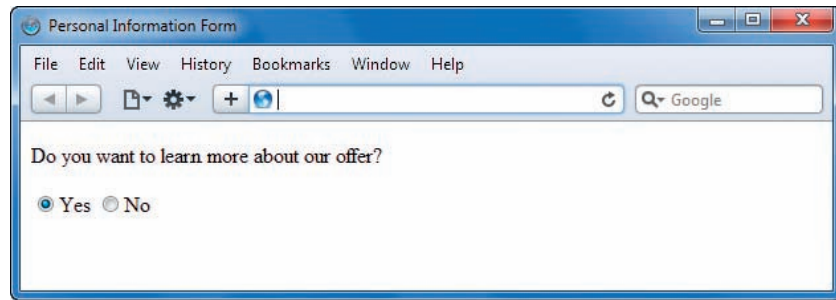


Figure 11-31 Form with radio buttons

4. In this project, you will build a text area form element.
 - a. In your HTML editor, open the file **blankform.html** in the Chapter11 folder in your work folder.
 - b. Save the file as **textarea.html** in the same location.
 - c. Examine the code. The file contains only the default HTML elements and an empty `<form>` element.
 - d. Build the form shown in Figure 11-32. The text area is 6 rows by 35 columns.

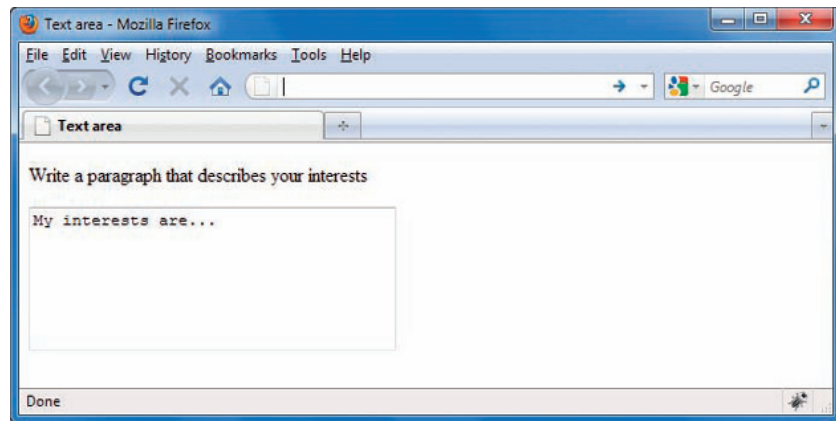


Figure 11-32 Form with text area

5. In this project, you will build a select form element.
 - a. In your HTML editor, open the file **blankform.html** in the Chapter11 folder in your work folder.
 - b. Save the file as **select.html** in the same location.
 - c. Examine the code. The file contains only the default HTML elements and an empty `<form>` element.

- d. Build the form shown in Figure 11-33.

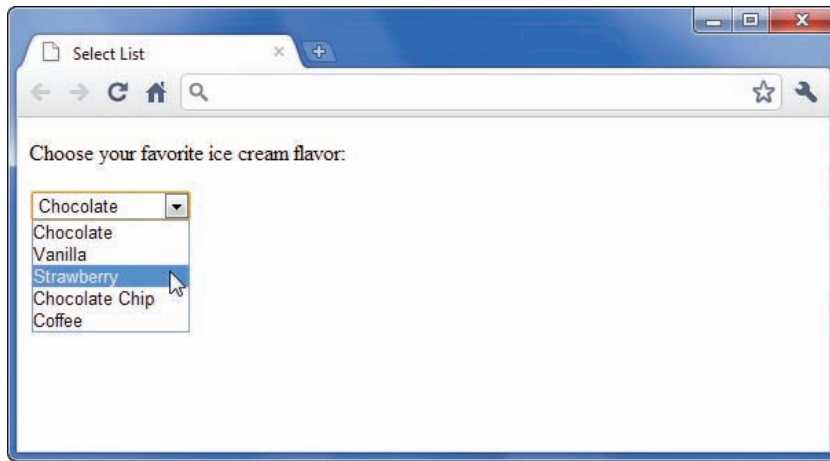
A screenshot of a web browser window titled "Select List". The browser's address bar is empty. The page content includes the text "Choose your favorite ice cream flavor:" followed by a list box. The list box is open, showing a dropdown menu with the following options: "Chocolate", "Chocolate", "Vanilla", "Strawberry", "Chocolate Chip", and "Coffee". A mouse cursor is pointing at the "Strawberry" option.

Figure 11-33 Form with list box

Individual Case Project

Build a user feedback form for your project Web site. You can refer to the sample feedback form in Chapter 3 for ideas. Customize the types of questions you ask to match the content of your site. Create both scaled questions and open-ended questions for your users. For example, ask users to rate the navigation of your site on a scale of 1 to 5, and also include a text area input where they write about their experience of navigating your Web site. Although you will not be able to activate the form (because you don't have an appropriate script to process the data), you can demonstrate the types of questions you would ask users to find out more about their habits when they visit your site.

Team Case Project

Each team member creates and submits to the instructor his or her own feedback form, as described in the preceding Individual Case Project. You are free to design the form any way you choose, but it must include the navigation characteristics, typographic specifications, and design conventions of your project Web site.

Then meet as a team and choose the best features, questions, and design characteristics from each member's submitted form. Create a new form that combines these features, and add it to your project Web site.