



Forms

Chapter Objectives

In this chapter, you will learn how to . . .

- Describe common uses of forms on web pages
- Create forms on web pages using the form, input, textarea, and select elements
- Create forms that provide additional accessibility features using the accesskey and tabindex attributes
- Associate form controls and groups using the label, fieldset, and legend elements
- Create custom image buttons and use the button element
- Use CSS to style a form
- Configure the new HTML5 form controls, including the e-mail address, URL, datalist, range, spinner, calendar, and color controls
- Describe the features and common uses of server-side processing
- Invoke server-side processing to handle form data
- Find free server-side processing resources on the Web

Forms are used for many purposes all over the Web. They are used by search engines to accept keywords and by online stores to process e-commerce shopping carts. Websites use forms to help with a variety of functions, including accepting visitor feedback, encouraging visitors to send a news story to a friend or colleague, collecting e-mail addresses for a newsletter, and accepting order information. This chapter introduces a very powerful tool for web developers—forms that accept information from web page visitors.

9.1 Overview of Forms

Every time you use a search engine, place an order, or join an online mailing list, you use a **form**. A form is an HTML element that contains and organizes objects called **form controls**, including text boxes, check boxes, and buttons, that can accept information from website visitors.

For example, you may have used Google's search form (<http://www.google.com>) many times but never thought about how it works. The form is quite simple; it contains just three form controls: the text box that accepts the keywords used in the search and two search buttons. The "Google Search" button submits the form and invokes a process to search the Google databases and display a results page. The whimsical "I'm Feeling Lucky" button submits the form and displays the top page for your keywords.

Figure 9.1 shows a form that is used to enter shipping information. This form contains text boxes to accept information such as name and address. Select lists (sometimes called drop-down boxes) are used to capture information with a limited number of correct values, such as state and country information.

When a visitor clicks the continue button, the form information is submitted and the ordering process continues.

Whether a form is used to search for web pages or to order a publication, the form alone cannot do all of the processing. The form needs to invoke a program or script on the web server in order to search a database or record an order. There are usually two components of a form:

1. The HTML form itself, which is the web page user interface
2. The server-side processing, which works with the form data and sends e-mail, writes to a text file, updates a database, or performs some other type of processing on the server



The image shows a web form titled "Shipping Address". It contains the following fields and controls:

- Name:** A text input field.
- Address Line 1:** A text input field.
- Address Line 2:** A text input field.
- City:** A text input field.
- State:** A dropdown menu with a downward arrow.
- Zip:** A text input field.
- Country:** A dropdown menu currently displaying "United States" with a downward arrow.
- Continue:** A button at the bottom left of the form.

Figure 9.1 This form accepts shipping information

Form Element

Now that you have a basic understanding of what forms do, let's focus on the HTML code to create a form. The **form element** contains a form on a web page. The `<form>` tag specifies the beginning of a form area. The closing `</form>` tag specifies the end of a form area. There can be multiple forms on a web page, but they cannot be nested inside one another. The form element can be configured with attributes that specify which server-side program or file will process the form, how the form information will be sent to the server, and the name of the form. These attributes are listed in Table 9.1.

Table 9.1 Attributes of the form element

| Attribute | Value | Purpose |
|--------------|--|--|
| action | URL or file name/path of server-side processing script | Required; indicates where to send the form information when the form is submitted; <code>mailto:e-mailaddress</code> will launch the visitor's default e-mail application to send the form information |
| autocomplete | <code>on</code> <code>off</code> | HTML5 attribute; default value; browser will use autocomplete to fill form fields HTML5 attribute; browser will not use autocomplete to fill form fields |
| id | Alphanumeric, no spaces; the value must be unique and not used for other id values on the same web page document | Optional; provides a unique identifier for the form |
| method | <code>get</code> <code>post</code> | Default value; the value of <code>get</code> causes the form data to be appended to the URL and sent to the web server The <code>post</code> method is more private and transmits the form data in the body of the HTTP response; this method is preferred by the W3C |
| name | Alphanumeric, no spaces, begins with a letter; choose a form name value that is descriptive but short (for example, <code>OrderForm</code> is better than <code>Form1</code> or <code>WidgetsRUsOrderForm</code>) | Optional; names the form so that it can be easily accessed by client-side scripting languages to edit and verify the form information before the server-side processing is invoked |

For example, to configure a form with the `name` attribute set to the value “order”, using the `post` method and invoking a script called `demo.php` on your web server, the code is

```
<form name="order" method="post" id="order" action="demo.php">
. . . form controls go here . . .
</form>
```

Form Controls

The purpose of a form is to gather information from a web page visitor. Form controls are the objects that accept the information. Types of form controls include text boxes, scrolling text boxes, select lists, radio buttons, check boxes, and buttons. HTML5 offers new form controls, including those that are customized for e-mail addresses, URLs, dates, times, numbers, and even date selection. HTML elements that configure form controls will be introduced in the following sections.

9.2 Input Element Form Controls

The **input element** is a stand-alone, or void, tag that is used to configure several different types of form controls. The input element is not coded as a pair of opening and closing tags. Use the `type` attribute to specify the type of form control that the browser should display.

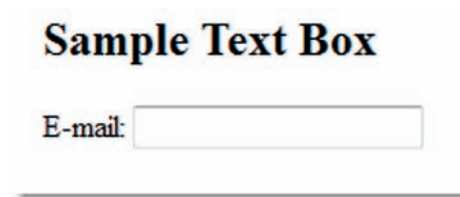


Figure 9.2 The `<input>` tag with `type="text"` configures this form element

Text Box

The `<input>` tag with `type="text"` configures a **text box** form control. The text box form control accepts text or numeric information such as names, e-mail addresses, phone numbers, and other text. Common input element attributes for text boxes are listed in Table 9.2. A text box is shown in Figure 9.2.

The code for the text box is shown below:

```
E-mail: <input type="text" name="email" id="email">
```

Table 9.2 Common text box attributes

| Attribute | Value | Purpose |
|---------------------------|---|---|
| <code>type</code> | <code>text</code> | Configures a text box |
| <code>name</code> | Alphanumeric, no spaces, begins with a letter | Names the form element so that it can be easily accessed by client-side scripting languages or by server-side processing; the name should be unique |
| <code>id</code> | Alphanumeric, no spaces, begins with a letter | Provides a unique identifier for the form element |
| <code>size</code> | Numeric value | Configures the width of the text box as displayed by the browser; if size is omitted, the browser displays the text box with its own default size |
| <code>maxlength</code> | Numeric value | Configures the maximum length of data accepted by the text box |
| <code>value</code> | Text or numeric characters | Assigns an initial value to the text box that is displayed by the browser; accepts information typed in the text box; this value can be accessed by client-side scripting languages and by server-side processing |
| <code>disabled</code> | <code>disabled</code> | Form control is disabled |
| <code>readonly</code> | <code>readonly</code> | Form control is for display; cannot be edited |
| <code>autocomplete</code> | <code>on</code> <code>off</code> | HTML5 attribute; default; browser will use autocomplete to fill the form control HTML5 attribute; browser will not use autocomplete to fill the form control |
| <code>autofocus</code> | <code>autofocus</code> | HTML5 attribute; browser places cursor in the form control and sets the focus |
| <code>list</code> | Datalist element id value | HTML5 attribute; associates the form control with a datalist element |
| <code>placeholder</code> | Text or numeric characters | HTML5 attribute; brief information intended to assist the user |
| <code>required</code> | <code>required</code> | HTML5 attribute; browser verifies entry of information before submitting the form |
| <code>accesskey</code> | Keyboard character | Configures a hot key for the form control |
| <code>tabindex</code> | Numeric value | Configures the tab order of the form control |

Several attributes are new in HTML5. The new **required attribute** is exciting because it will cause supporting browsers to perform form validation. Browsers that support the HTML5 required attribute will automatically verify that information has been entered in the text box and display an error message when the condition is not met. A code sample is

```
E-mail: <input type="text" name="email" id="email"
required="required">
```

Figure 9.3 shows an error message automatically generated by Firefox that is displayed after the user clicked the form's submit button without entering information in the required text box. Browsers that do not support HTML5 or the required attribute will ignore the attribute.

Although web designers are enthusiastic about the required attribute and other new form-processing functions offered by HTML5, it will be some time before all browsers support these new features. In the meantime, be aware that verification and validation of form information also must be done the old-fashioned way—with client-side or server-side scripting.

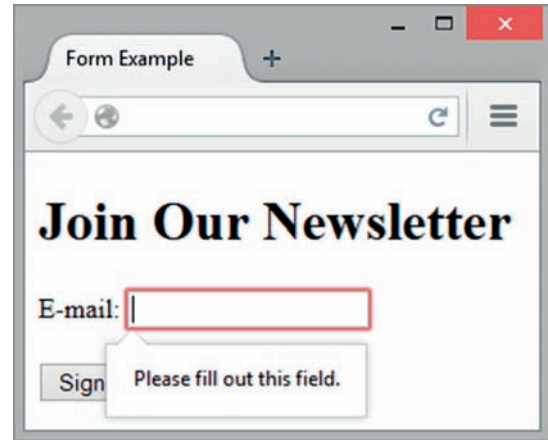


Figure 9.3 The Firefox browser displayed an error message



FAQ Why use both the name and id attributes on form controls?

The **name attribute** names the form element so that it can be easily accessed by client-side scripting languages such as JavaScript and by server-side processing languages such as PHP. The value given to a `name` attribute for a form element should be unique for that form. The `id` attribute is included for use with CSS and access by client-side scripting languages such as JavaScript. The value of the `id` attribute should be unique to the entire web page document that contains the form. Typically, the values assigned to the `name` and `id` attributes on a particular form element are the same.

Submit Button

The **submit button** form control is used to submit the form. When clicked, it triggers the action method on the form element and causes the browser to send the form data (the name and value pairs for each form control) to the web server. The web server will invoke the server-side processing program or script listed on the form's action property.

The input element with `type="submit"` configures a submit button. For example,

```
<input type="submit">
```

Reset Button

The **reset button** form control is used to reset the form fields to their initial values. A reset button does not submit the form.

The input element with `type="reset"` configures a reset button. For example,

```
<input type="reset">
```

A form with a text box, a submit button, and a reset button is shown in Figure 9.4.

Sample Form

E-mail:

Figure 9.4 This form contains a text box, a submit button, and a reset button

Common attributes for submit buttons and reset buttons are listed in Table 9.3.

Table 9.3 Common attributes for submit and reset buttons

| Attribute | Value | Purpose |
|-----------|---|--|
| type | submit | Configures a submit button |
| | reset | Configures a reset button |
| name | Alphanumeric, no spaces, begins with a letter | Names the form element so that it can be easily accessed by client-side scripting languages (such as JavaScript) or by server-side processing; the name should be unique |
| id | Alphanumeric, no spaces, begins with a letter | Provides a unique identifier for the form element |
| value | Text or numeric characters | Configures the text displayed on the button; a submit button displays the text “Submit Query” by default; a reset button displays “Reset” by default |
| accesskey | Keyboard character | Configures a hot key for the form control |
| tabindex | Numeric value | Configures the tab order of the form control |



Hands-On Practice 9.1

You will code a form in this Hands-On Practice. To get started, launch a text editor and open `chapter2/template.html` in the student files. Save the file as `form1.html`. You will create a web page with a form similar to the example in Figure 9.5.

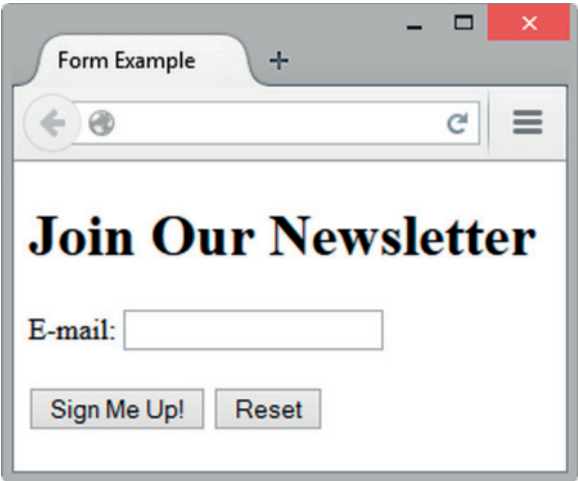


Figure 9.5 The text on the submit button says “Sign Me Up!”

1. Modify the title element to display the text “Form Example”.
2. Configure an `h1` element with the text “Join Our Newsletter”.

You are ready to configure the form area. A form begins with the form element. Insert a blank line under the heading you just added and type in a `<form>` tag as follows:

```
<form method="get">
```

In your first form, we are using the minimal HTML needed to create the form; we’ll begin working with the `action` attribute later in the chapter.

3. To create the form control for the visitor's e-mail address to be entered, type the following code on a blank line below the form element:

```
E-mail: <input type="text" name="email" id="email"><br><br>
```

This places the text “E-mail:” in front of the text box used to enter the visitor's e-mail address. The input element has a `type` attribute with the value of `text` that causes the browser to display a text box. The `name` attribute assigns the name `e-mail` to the information entered into the text box (the `value`) and could be used by server-side processing. The `id` attribute uniquely identifies the element on the page. The `
` elements configure line breaks.

4. Now you are ready to add the submit button to the form on the next line. Add a `value` attribute set to “Sign Me Up!”:

```
<input type="submit" value="Sign Me Up!">
```

This causes the browser to display a button with “Sign Me Up!” instead of the default value of “Submit Query”.

5. Add a blank space after the submit button and code a reset button:

```
<input type="reset">
```

6. Next, code the closing form tag:

```
</form>
```

Save `form1.html` and test your web page in a browser. It should look similar to the page shown in Figure 9.5.

You can compare your work with the solution found in the student files (`chapter9/9.1/form.html`). Try entering some information into your form. Try clicking the submit button. Don't worry if the form redisplay but nothing seems to happen when you click the button—you haven't configured this form to work with any server-side processing. Connecting forms to server-side processing is demonstrated later in this chapter. The next sections will introduce you to more form controls.

Check Box

The **check box** form control allows the user to select one or more of a group of predetermined items. The input element with `type="checkbox"` configures a check box. Common check box attributes are listed in Table 9.4.

Figure 9.6 shows an example with several check boxes. Note that more than one check box can be selected by the user. The HTML is

```
Choose the browsers you use: <br>
<input type="checkbox" name="IE" id="IE" value="yes">
Internet Explorer<br>
<input type="checkbox" name="Firefox" id="Firefox"
value="yes"> Firefox<br>
<input type="checkbox" name="Opera" id="Opera"
value="yes"> Opera<br>
```

Sample Check Box

Choose the browsers you use:

☐ Internet Explorer

☐ Firefox

☐ Opera

Figure 9.6 Sample check box

Table 9.4 Common check box attributes

| Attribute | Value | Purpose |
|-----------|---|--|
| type | checkbox | Configures a check box |
| name | Alphanumeric, no spaces, begins with a letter | Names the form element so that it can be easily accessed by client-side scripting languages (such as JavaScript) or by server-side processing; the name should be unique |
| id | Alphanumeric, no spaces, begins with a letter | Provides a unique identifier for the form element |
| checked | checked | Configures the check box to be checked by default when displayed by the browser |
| value | Text or numeric characters | Assigns a value to the check box that is triggered when the check box is checked; this value can be accessed by client-side and server-side processing |
| disabled | disabled | Form control is disabled |
| readonly | readonly | Form control is for display; cannot be edited |
| autofocus | autofocus | HTML5 attribute; browser places cursor in the form control and sets the focus |
| required | required | HTML5 attribute; browser verifies entry of information before submitting the form |
| accesskey | Keyboard character | Configures a hot key for the form control |
| tabindex | Numeric value | Configures the tab order of the form control |

Sample Radio Button

Select your favorite browser:

- ☐ Internet Explorer
☐ Firefox
☐ Opera

Figure 9.7 Use radio buttons when only one choice is an appropriate response

Radio Button

The **radio button** form control allows the user to select exactly one (and only one) choice from a group of predetermined items. Each radio button in a group is given the same `name` attribute and a unique `value` attribute. Because the name attribute is the same, the elements are identified as part of a group by the browsers and only one may be selected.

The input element with `type="radio"` configures a radio button. Figure 9.7 shows an example with a radio button group. Note that only one radio button can be selected at a time by the user. Common radio button attributes are listed in Table 9.5. The HTML is

```
Select your favorite browser:<br>
<input type="radio" name="favbrowser" id="favIE" value="IE"> Internet
Explorer<br>
<input type="radio" name="favbrowser" id="favFirefox" value="Firefox">
Firefox<br>
<input type="radio" name="favbrowser" id="favOpera" value="Opera">
Opera<br>
```

Notice that all the `name` attributes have the same value: `favbrowser`. Radio buttons with the same `name` attribute are treated as a group by the browser. Each radio button in the same group can be uniquely identified by its `value` attribute. Common radio button attributes are listed in Table 9.5.

Table 9.5 Common radio button attributes

| Attribute | Value | Purpose |
|-----------|---|---|
| type | radio | Configures a radio button |
| name | Alphanumeric, no spaces, begins with a letter | Names the form element so that it can be easily accessed by client-side scripting languages or by server-side processing |
| id | Alphanumeric, no spaces, begins with a letter | Provides a unique identifier for the form element |
| checked | checked | Configures the radio button to be selected by default when displayed by the browser |
| value | Text or numeric characters | Assigns a value to the radio button that is triggered when the radio button is selected; this should be a unique value for each radio button in a group |
| disabled | disabled | Form control is disabled |
| readonly | readonly | Form control is for display; cannot be edited |
| autofocus | autofocus | HTML5 attribute; browser places cursor in the form control and sets the focus |
| required | required | HTML5 attribute; browser verifies entry of information before submitting the form |
| accesskey | Keyboard character | Configures a hot key for the form control |
| tabindex | Numeric value | Configures the tab order of the form control |

Hidden Input Control

The **hidden input control** stores text or numeric information, but it is not visible in the browser viewport. Hidden controls can be accessed by both client-side scripting and server-side processing.

The input element with `type="hidden"` configures a hidden input control. Common attributes for hidden input controls are listed in Table 9.6. The HTML to create a hidden input control with the `name` attribute set to “sendto” and the `value` attribute set to an e-mail address is

```
<input type="hidden" name="sendto" id="sendto" value="order@site.com">
```

Table 9.6 Common hidden input control attributes

| Attribute | Value | Purpose |
|-----------|---|---|
| type | hidden | Configures a hidden element |
| name | Alphanumeric, no spaces, begins with a letter | Names the form element so that it can be easily accessed by client-side scripting languages or by server-side processing; the name should be unique |
| id | Alphanumeric, no spaces, begins with a letter | Provides a unique identifier for the form element |
| value | Text or numeric characters | Assigns a value to the hidden control; this value can be accessed by client-side scripting languages and server-side processing |
| disabled | disabled | Form control is disabled |

Password Box

The **password box** form control is similar to the text box, but it is used to accept information that must be hidden as it is entered, such as a password.



Figure 9.8 The characters secret9 were typed, but the browser does not display them. (Note: Your browser may use a different symbol, such as a stylized circle, to hide the characters.)

The input element with `type="password"` configures a password box. When the user types information in a password box, asterisks (or another symbol, depending on the browser) are displayed instead of the characters that have been typed, as shown in Figure 9.8. This hides the information from someone looking over the shoulder of the person typing. The actual characters typed are sent to the server and the information is not really secret or hidden. See Chapter 12 for a discussion of encryption and security.

A password box is a specialized text box. See Table 9.2 for a list of text box attributes.

The HTML is

```
Password: <input type="password" name="pword" id="pword">
```

9.3 Scrolling Text Box

Textarea Element

The **scrolling text box** form control accepts free-form comments, questions, or descriptions. The **textarea element** configures a scrolling text box. The `<textarea>` tag denotes the beginning of the scrolling text box. The closing `</textarea>` tag denotes the end of the scrolling text box. Text contained between the tags will display in the scrolling text box area. A sample scrolling text box is shown in Figure 9.9.



Figure 9.9 Scrolling text box

Common attributes for scrolling text boxes are listed in Table 9.7. The HTML is

```
Comments:<br>
<textarea name="comments" id="comments" cols="40" rows="2"> Enter your
comments here</textarea>
```

Table 9.7 Common scrolling text box attributes

| Attribute | Value | Purpose |
|-------------|---|---|
| name | Alphanumeric, no spaces, begins with a letter | Names the form element so that it can be easily accessed by client-side scripting languages or by server-side processing; the name should be unique |
| id | Alphanumeric, no spaces, begins with a letter | Provides a unique identifier for the form element |
| cols | Numeric value | Required; configures the width in character columns of the scrolling text box; if cols is omitted, the browser displays the scrolling text box with its own default width |
| rows | Numeric value | Required; configures the height in rows of the scrolling text box; if rows is omitted, the browser displays the scrolling text box with its own default height |
| maxlength | Numeric value | Configures the maximum length of data accepted by the text box |
| disabled | disabled | Form control is disabled |
| readonly | readonly | Form control is for display; cannot be edited |
| autofocus | autofocus | HTML5 attribute; browser places cursor in the form control and sets the focus |
| placeholder | Text or numeric characters | HTML5 attribute; brief information intended to assist the user |
| required | required | HTML5 attribute; browser verifies entry of information before submitting the form |
| wrap | hard or soft | HTML5 attribute; configures line breaks within the information entered |
| accesskey | Keyboard character | Configures a hot key for the form control |
| tabindex | Numeric value | Configures the tab order of the form control |



Hands-On Practice 9.2

In this Hands-On Practice, you will create a contact form (see Figure 9.10) with the following form controls: a First Name text box, a Last Name text box, an E-mail text box, and a Comments scrolling text box. You'll use the form you created in Hands-On Practice 9.1 (see Figure 9.5) as a starting point. Launch a text editor and open `chapter9/9.1/form.html` in the student files. Save the file as `form2.html`.

Figure 9.10 A typical contact form

1. Modify the title element to display the text “Contact Form”.
2. Configure the h1 element with the text “Contact Us”.
3. A form control for the e-mail address is already coded. Refer to Figure 9.10 and note that you’ll need to add text box form controls for the first name and last name above the e-mail form control. Add the following code on new lines below the opening form tag to accept the name of your web page visitor:

```
First Name: <input type="text" name="fname" id="fname"><br><br>
Last Name: <input type="text" name="lname" id="lname"><br><br>
```

4. Now you are ready to add the scrolling text box form control to the form using a `<textarea>` tag on a new line below the e-mail form control. The code is

```
Comments:<br>
<textarea name="comments" id="comments"></textarea><br><br>
```

5. Save your file and display your web page in a browser to view the default display of a scrolling text box. Note that this default display will differ by browser. At the time this was written, Internet Explorer always rendered a vertical scroll bar, but the Firefox browser only rendered scroll bars once enough text was entered to require them. The writers of browser rendering engines keep the lives of web designers interesting!
6. Let’s configure the `rows` and `cols` attributes for the scrolling text box form control. Modify the `<textarea>` tag and set `rows="4"` and `cols="40"` as follows:

```
Comments:<br>
<textarea name="comments" id="comments" rows="4"
cols="40"></textarea><br><br>
```

7. Next, modify the text displayed on the submit button. Set the value attribute to “Contact”. Save form2.html and test your web page in a browser. It should look similar to the page shown in Figure 9.10.

You can compare your work with the solution found in the student files (chapter9/9.2/form.html). Try entering some information into your form. Try clicking the submit button. Don’t worry if the form redisplay but nothing seems to happen when you click the button—you haven’t configured this form to work with any server-side processing. Connecting forms to server-side processing is demonstrated later in this chapter.



FAQ How can I send form information in an e-mail?

Forms usually need to invoke some type of server-side processing to perform functions such as sending e-mail, writing to text files, updating databases, and so on. Another option is to set up a form to send information using the e-mail program configured to work with the web page visitor’s browser. In what is sometimes called using a mailto: URL, the `<form>` tag is coded to use your e-mail address in the action attribute:

```
<form method="post" action="mailto:me@webdevfoundations.net">
```

When a form is used in this manner, the web visitor will see a warning message. The warning message presents a nonprofessional image and is not the best way to inspire trust and confidence in your website or business.

Be aware that information sent in e-mail messages is not secure. Sensitive information, such as credit card numbers, should not be transmitted using e-mail. See Chapter 12 for information about using encryption to transmit data securely.

There are other reasons not to use `mailto: URL`. For example, when people share a computer, they may not be using the default e-mail application. In this case, filling out the form is a waste of time. Even if the person using the computer also uses the default e-mail application, perhaps he or she may not want to divulge this particular e-mail address. Perhaps they have another e-mail address that is used for forms and newsletters, and do not want to waste time filling out your form. In either case, the result is an unhappy website visitor. So, while using `mailto: URL` is easy, it does not always create the most usable web form for your visitors. What's a web developer to do? Use server-side processing (see Hands-On Practice 9.5) to handle form data instead of `mailto: URL`.

9.4 Select List

The **select list** form control shown in Figures 9.11 and 9.12 is also known by several other names, including select box, drop-down list, drop-down box, and option box. A select list is configured with one select element and multiple option elements.

Select Element

The **select element** contains and configures the select list form control. The `<select>` tag denotes the beginning of the select list. The closing `</select>` tag denotes the end of the select list. Attributes configure the number of options to display and whether more than one option item may be selected. Common attributes for select elements are listed in Table 9.8.

Table 9.8 Common select element attributes

| Attribute | Value | Purpose |
|-----------------------|---|--|
| <code>name</code> | Alphanumeric, no spaces, begins with a letter | Names the form element so that it can be easily accessed by client-side scripting languages or by server-side processing; the name should be unique |
| <code>id</code> | Alphanumeric, no spaces, begins with a letter | Provides a unique identifier for the form element |
| <code>size</code> | Numeric value | Configures the number of choices the browser will display; if set to 1, the element functions as a drop-down list (see Figure 9.12); scroll bars are automatically added by the browser if the number of options exceeds the space allowed |
| <code>multiple</code> | <code>multiple</code> | Configures a select list to accept more than one choice; by default, only one choice can be made from a select list |
| <code>disabled</code> | <code>disabled</code> | Form control is disabled |
| <code>tabindex</code> | Numeric value | Configures the tab order of the form control |

Option Element

The **option element** contains and configures an option item displayed in the select list form control. The `<option>` tag denotes the beginning of the option item. The closing `</option>` tag denotes the end of the option item. Attributes configure the value of the option and whether they are preselected. Common attributes for option elements are listed in Table 9.9.

Table 9.9 Common option element attributes

| Attribute | Value | Purpose |
|-----------|----------------------------|---|
| value | Text or numeric characters | Assigns a value to the option; this value can be accessed by client-side and server-side processing |
| selected | selected | Configures an option to be initially selected when displayed by a browser |
| disabled | disabled | Form control is disabled |

The HTML for the select list in Figure 9.11 is

```
<select size="1" name="favbrowser" id="favbrowser">
  <option>Select your favorite browser</option>
  <option value="Internet Explorer">Internet Explorer</option>
  <option value="Firefox">Firefox</option>
  <option value="Opera">Opera</option>
</select>
```

The HTML for the select list in Figure 9.12 is

```
<select size="4" name="jumpmenu" id="jumpmenu">
  <option value="index.html">Home</option>
  <option value="products.html">Products</option>
  <option value="services.html">Services</option>
  <option value="about.html">About</option>
  <option value="contact.html">Contact</option>
</select>
```

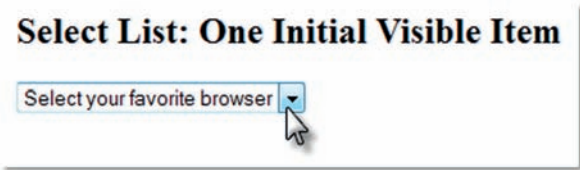


Figure 9.11 A select list with size set to 1 functions as a drop-down box when the arrow is clicked



Figure 9.12 Because there are more than four choices, the browser displays a scroll bar



FAQ How does the menu in Figure 9.12 display the selected page?

Well, it doesn't work, yet. It needs JavaScript (see Chapter 14) to check for the selected item and direct the browser to display the new document. Because it requires JavaScript to work, this type of menu would not be a good choice for your main navigation, but it might be useful for a secondary navigation area.



Checkpoint 9.1

1. You are designing a website for a client who sells items in a retail store. They want to create a customer list for e-mail marketing purposes. Your client sells to consumers and needs a form that accepts a customer's name and e-mail address. Would you recommend using two input boxes (one for the name and one for the e-mail) or three input boxes (one each for the first name, last name, and e-mail address)? Explain your answer.
2. A question on a survey asks participants to indicate their favorite browsers. Most people will select more than one response. What type of form control would you use to configure this question on the web page? Explain your answer.
3. True or False? In a radio button group, the **value attribute** is used by the browser to process the separate radio buttons as a group.

9.5 Image Buttons and the Button Element

As you have worked with forms in this chapter, you may have noticed that the standard submit button (see Figure 9.10) is a little plain. You can make the form control that visitors select to submit the form a bit more compelling and visually interesting in two ways:

1. Configure an image with the input element.
2. Create a custom image that is configured with the button element.

Image Button

Figure 9.13 shows an image used in place of the standard submit button. This is called an **image button**. When an image button is clicked or tapped, the form is submitted. The image button is coded using the `<input>` tag along with `type="image"` and a `src` attribute with the value of the name of the image file. For example, to use the image called `login.gif` as an image button, the HTML code is

```
<input type="image" src="login.gif" alt="Login Button">
```

Figure 9.13 The web page visitor will select the image button to submit the form

Button Element

Another way to add more interest to a form is to use the **button element**, which can be used to configure not only images but also blocks of text as the selectable area that can submit or reset a form. Any web page content that is between the `<button>` and `</button>` tags is configured to be part of the button. Common attributes for button elements are listed in Table 9.10.

Figure 9.14 shows a form that has an image (`signup.gif`) configured as a submit button using the button element.

Figure 9.14 The button element configured as a submit button

Table 9.10 Common button element attributes

| Common Attributes | Values | Purpose |
|-------------------|---|---|
| type | submit | Functions as a submit button |
| | reset | Functions as a reset button |
| | button | Functions as a button |
| name | Alphanumeric, no spaces, begins with a letter | Names the form element so that it can be easily accessed by client-side scripting languages or by server-side processing; the name should be unique |
| id | Alphanumeric, no spaces, begins with a letter | Provides a unique identifier for the form element |
| alt | Brief text description of the image | Provides accessibility to visitors who are unable to view the image |
| value | Text or numeric characters | A value given to a form element that is passed to the form handler |

The following HTML code creates the button shown in Figure 9.14:

```
<button type="submit">
<br>Sign up for free newsletter
</button>
```

As you visit web pages and view their source code, you will find that the button element is not used as often as the standard submit button or the image button.

9.6 Accessibility and Forms

Focus on Accessibility



In this section, you'll explore techniques to increase the accessibility of form controls, including the label element, fieldset element, legend element, tabindex attribute, and accesskey attribute, which make it easier for individuals with vision and mobility challenges to use your form pages. The use of label, fieldset, and legend elements may increase the readability and usability of the web form for all visitors.

Label Element

Focus on Accessibility



The **label element** is a container tag that associates a text description with a form control. This is helpful to visually challenged individuals who are using assistive technology such as a screen reader to match up the text descriptions on forms with their corresponding form controls. The label element also benefits individuals who have difficulty with fine motor control. Clicking anywhere on either a form control or its associated text label will set the cursor focus to the form control. The **<label>** tag specifies the beginning of the label. The closing **</label>** tag specifies the end of the label.

There are two different methods to associate a label with a form control.

1. The first method places the label element as a container around both the text description and the HTML form element. Notice that both the text label and the form control must be adjacent elements. The code is

```
<label>E-mail: <input type="text" name="email" id="email"></label>
```


2. The second method uses the `for` attribute to associate the label with a particular HTML form element. This is more flexible and does not require the text label and the form control to be adjacent. The code is

```
<label for="email">E-mail: </label>
<input type="text" name="email" id="email">
```

Notice that the value of the **for** attribute on the label element is the same as the value of the `id` attribute on the input element. This creates the association between the text label and the form control. The input element uses both the `name` and `id` attributes for different purposes. The `name` attribute can be used by client-side scripting and server-side processing. The `id` attribute creates an identifier that can be used by the label element, anchor element, and CSS selectors. The label element does not display on the web page—it works behind the scenes to provide for accessibility.



Hands-On Practice 9.3

In this Hands-On Practice, you will add the label element to the text box and scrolling text area form controls on the form you created in Hands-On Practice 9.2 (see Figure 9.10) as a starting point. Launch a text editor and open `chapter9/9.2/form.html` in the student files. Save the file as `form3.html`.

1. Locate the text box for the first name. Add a label element to wrap around the input tag as follows:

```
<label>First Name: <input type="text" name="fname"
id="fname"></label>
```

2. In a similar manner, add a label element for the last name and the e-mail form controls.
3. Configure a label element to contain the text “Comments”. Associate the label with the scrolling text box form control. The sample code is

```
<label for="comments">Comments:</label><br>
<textarea name="comments" id="comments" rows="4"
cols="40"></textarea>
```

Save `form3.html` and test your web page in a browser. It should look similar to the page shown in Figure 9.10—the label elements do not change the way that the page displays, but a web visitor with physical challenges should find the form easier to use.

You can compare your work with the solution found in the student files (`chapter9/9.3/form.html`). Try entering some information into your form. Try clicking the submit button. Don’t worry if the form redisplayed but nothing seems to happen when you click the button—you haven’t configured this form to work with any server-side processing. Connecting forms to server-side processing is demonstrated later in this chapter.

Fieldset and Legend Elements

A technique that can be used to create a more visually pleasing form is to group elements of a similar purpose together using the **fieldset element**, which will cause the browser to render a visual cue, such as an outline or a border, around form elements grouped together within the fieldset. The `<fieldset>` tag denotes the beginning of the grouping. The closing `</fieldset>` tag denotes the end of the grouping.

The **legend element** provides a text description for the fieldset grouping. The `<legend>` tag denotes the beginning of the text description. The closing `</legend>` tag denotes the end of the text description. The HTML to create the grouping shown in Figure 9.15 is

```
<fieldset>
<legend>Billing Address</legend>
<label>Street: <input type="text" name="street" id="street"
      size="54"></label><br><br>
<label>City: <input type="text" name="city" id="city"></label>
<label>State: <input type="text" name="state" id="state" maxlength="2"
      size="5"></label>
<label>Zip: <input type="text" name="zip" id="zip" maxlength="5"
      size="5"></label>
</fieldset>
```

Fieldset and Legend

Figure 9.15 Form controls that are all related to a mailing address

Focus on Accessibility



The grouping and visual effect of the fieldset element creates an organized and appealing web page containing a form. Using the fieldset and legend elements to group form controls enhances accessibility by organizing the controls both visually and semantically. The fieldset and legend elements can be accessed by screen readers and are useful tools for configuring groups of radio buttons and check boxes on web pages.



Hands-On Practice 9.4

In this Hands-On Practice, you will modify the contact form (form3.html) you worked with in Hands-On Practice 9.3 to use the fieldset and legend elements (see Figure 9.16).

Launch a text editor and open chapter9/9.3/form.html in the student files. Save the file as form4.html. Perform the following edits:

1. Add an opening `<fieldset>` tag after the opening `<form>` tag.
2. Immediately after the opening `<fieldset>` tag, code a legend element that contains the following text: "Customer Information".

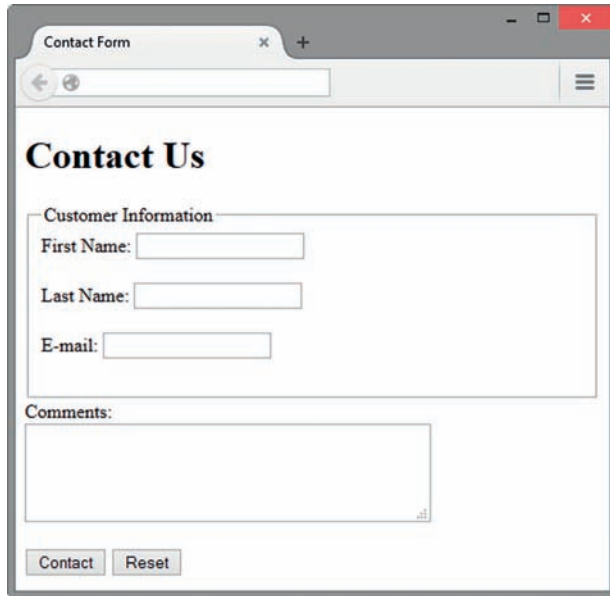


Figure 9.16 shows a web browser window titled "Contact Form" displaying a form titled "Contact Us". The form contains a "Customer Information" fieldset with three input fields: "First Name:", "Last Name:", and "E-mail:". Below this is a "Comments:" label followed by a large text area. At the bottom are "Contact" and "Reset" buttons.

Figure 9.16 The fieldset, legend, and label elements

3. Code the closing `</fieldset>` tag before the label element for the Comments scrolling text box.
4. Save your file and test your web page in a browser. It should look similar to the one shown in Figure 9.16. You can compare your work with the solution found in the student files (chapter9/9.4/form4.html). You may notice that when you activate the submit button, the form redisplay. This is because there is no action property in the form element. You'll work with setting the action property in Section 9.8.
5. How about a quick preview of styling a form with CSS? Figures 9.16 and 9.17 show the same form elements, but the form in Figure 9.17 is styled with CSS, which gives it the same functionality with increased visual appeal.

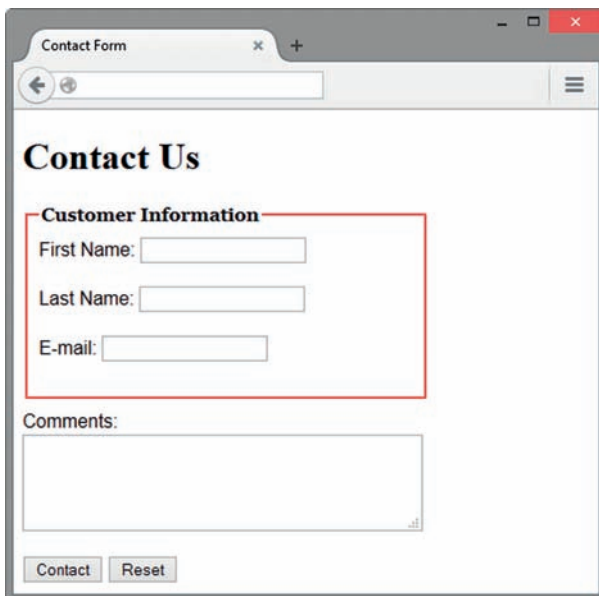


Figure 9.17 shows the same "Contact Form" as Figure 9.16, but with CSS styling. A red border is applied to the "Customer Information" fieldset, and the "Comments:" label is now bolded. The "Contact" and "Reset" buttons are also styled with a gray background and white text.

Figure 9.17 The fieldset, legend, and label elements are configured with CSS

Open form4.html in a text editor and add embedded styles to the head section as indicated below:

```
fieldset { width: 320px;
  border: 2px ridge #ff0000;
  padding: 10px;
  margin-bottom: 10px; }
legend { font-family: Georgia, "Times New Roman", serif;
  font-weight: bold; }
label { font-family: Arial, sans-serif; }
```

Save your file as form5.html and test your web page in a browser. It should look similar to the one shown in Figure 9.17. You can compare your work with the solution found in the student files (chapter9/9.4/form5.html).

Focus on Accessibility



The tabindex Attribute

Some of your website visitors may have difficulty using a mouse and will access your form with a keyboard. The Tab key can be used to move from one form control to another. The default action for the Tab key within a form is to move to the next form control in the order in which the form controls are coded in the web page document. This is usually appropriate. However, if the tab order needs to be changed for a form, use the **tabindex attribute** on each form control.

For each form tag (<input>, <select>, and <textarea>), code a **tabindex** attribute with a numeric value, beginning with 1, 2, 3, and so on in numerical order. The HTML code to configure the customer e-mail text box as the initial position of the cursor is

```
<input type="text" name="Email" id="Email" tabindex="1">
```

If you configure a form control with **tabindex="0"**, it will be visited after all of the other form controls that are assigned a **tabindex** attribute. If you happen to assign two form controls the same **tabindex** value, the one that is coded first in the HTML will be visited first.

You can configure the **tabindex** attribute for anchor tags in a similar manner. The default action for the Tab key and anchor tags is to move from hyperlink to hyperlink in the order they are coded on the page. Use the **tabindex** attribute if you need to modify this behavior.

Focus on Accessibility



The accesskey Attribute

Another technique that can make your form keyboard-friendly is the use of the **accesskey attribute** on form controls. You can also configure the **accesskey** attribute on an anchor tag. Assigning the **accesskey** attribute a value of one of the characters (a letter or number) on the keyboard will create a hot key that your website visitor can press to move the cursor immediately to a form control or hyperlink.

The method used to access this hot key varies depending on the operating system. Windows users will press the Alt key and the character key. Mac users will press the Ctrl key and the character key. For example, if the form shown in Figure 9.10 had the

customer e-mail text coded with `accesskey="E"`, the web page visitor using Windows could press the Alt and E keys to move the cursor immediately to the e-mail text box. The HTML code for this is

```
<input type="text" name="email" id="email" accesskey="E">
```

Note that you cannot rely on the browser to indicate that a character is an access key, also called a hot key. You will have to manually code information about the hot key. A visual cue may be helpful, such as displaying the hot key in bold or by placing a message such as (Alt+E) after a form control or hyperlink that uses a hot key. When choosing accesskey values, avoid combinations that are already used by the operating system (such as Alt+F to display the File menu). Testing hot keys is crucial.



Checkpoint 9.2

1. Describe the purpose of the fieldset and legend elements.
2. Describe the purpose of the `accesskey` attribute and how it supports accessibility.
3. When designing a form, should you use the standard submit button, an image button, or a button tag? Are these different in the way in which they provide for accessibility? Explain your answer.

9.7 Style a Form with CSS

The form in Figure 9.10 (from Hands-On Practice 9.2) looks a little “messy” and you might be wondering how that can be improved. In the time before CSS was well supported by browsers, web designers always used a table to configure the design of form elements, typically placing the text labels and form field elements in separate table data cells. However, the table approach is outdated, does not provide support for accessibility, and can be difficult to maintain over time. The modern approach is to style the form with CSS.

When styling a form with CSS, the box model is used to create a series of boxes, as shown in Figure 9.18. The outermost box defines the form area. Other boxes indicate label elements and form controls. CSS is used to configure these components.

Figure 9.19 displays a web page with a form configured in this manner (see `chapter9/formcss.html` in the student files). As you view the following CSS and HTML, note that the label element selector is configured with a 100 pixel width, floats to the left side of the form, and clears any previous left floats. The input and textarea elements have a top margin and are configured with block display. The submit button is assigned to an id with a left margin. The styles result in a well-aligned form.

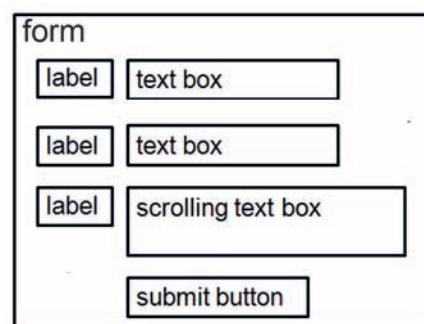


Figure 9.18 Wireframe for a form

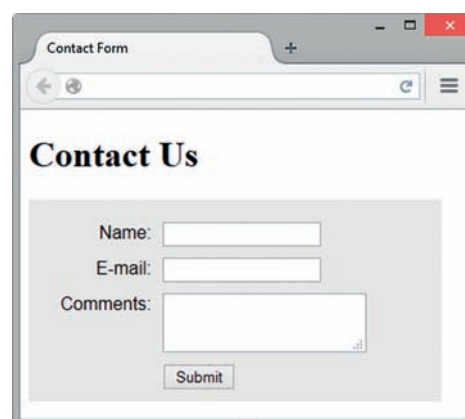


Figure 9.19 This form is configured with CSS

The CSS is

```
form { background-color:#eaeaea;
        font-family: Arial, sans-serif;
        padding: 10px; }
label { float: left;
        width: 100px;
        clear: left;
        text-align: right;
        padding-right: 10px;
        margin-top: 10px; }
input, textarea { margin-top: 10px;
                  display: block; }
#mySubmit { margin-left: 110px; }
```

The HTML code is

```
<form>
  <label for="myName">Name:</label>
  <input type="text" name="myName" id="myName">
  <label for="myEmail">E-mail:</label>
  <input type="text" name="myEmail" id="myEmail">
  <label for="myComments">Comments:</label>
  <textarea name="myComments" id="myComments" rows="2"
    cols="20"></textarea>
  <input id="mySubmit" type="submit" value="Submit">
</form>
```

This section provided you with a method to style a form with CSS. Testing the way that different browsers render the form is crucial.

As you've coded and displayed the forms in this chapter, you may have noticed that when you click the submit button, the form just redisplay—the form doesn't do anything. This is because there is no action attribute in the `<form>` tag. The next section focuses on the second component of using forms on web pages—server-side processing.

9.8 Server-Side Processing

Your web browser requests web pages and their related files from a web server. The web server locates the files and sends them to your web browser. Then the web browser renders the returned files and displays the requested web pages. Figure 9.20 illustrates the communication between the web browser and the web server.

Sometimes a website needs more functionality than static web pages, possibly a site search, order form, e-mail list, database display, or other type of interactive, dynamic processing. This is when server-side processing is needed. Early web servers used a protocol called **Common Gateway Interface (CGI)** to provide this functionality. CGI is a protocol, or standard method, for a web server to pass a web page user's request (which is typically initiated through the use of a form) to an application program and to accept information

**Figure 9.20**

The web browser (client) communicates with the web server

to send to the user. The web server typically passes the form information to a small application program that is run by the operating system and that processes the data, usually sending back a confirmation web page or message. Perl and C are popular programming languages for CGI applications.

Server-side scripting is a technology by which a server-side script is run on a web server to dynamically generate web pages. Examples of server-side scripting technologies include PHP, Ruby on Rails, Microsoft Active Server Pages, Adobe ColdFusion, Oracle JavaServer Pages, and Microsoft .NET. Server-side scripting differs from CGI in that it uses **direct execution**. The script is run either by the web server itself or by an extension module to the web server.

A web page invokes server-side processing by either an attribute on a form or by a hyperlink (the URL of the script is used). Any form data that exists is passed to the script. The script completes its processing and may generate a confirmation or response web page with the requested information. When invoking a server-side script, the web developer and the server-side programmer must communicate about the form **method attribute** (`get` or `post`), form **action attribute** (the URL of the server-side script), and any special form element control(s) expected by the server-side script.

The `method` attribute is used on the form tag to indicate the way in which the name and value pairs should be passed to the server. The method attribute value of `get` causes the form data to be appended to the URL, which is easily visible and not secure. The method attribute value of `post` does not pass the form information in the URL; it passes it in the entity body of the HTTP request, which makes it more private. The W3C recommends the `method="post"` method.

The `action` attribute is used on the `<form>` tag to invoke a server-side script. The `name` attribute and the `value` attribute associated with each form control are passed to the server-side script. The `name` attribute may be used as a variable name in the server-side processing. In the next Hands-On Practice, you will invoke a server-side script from a form.



VideoNote

**Connect a Form to
Server-Side Processing**



Hands-On Practice 9.5

In this Hands-On Practice, you will configure a form to invoke a server-side script. Please note that your computer must be connected to the Internet when you test your work. When using a server-side script, you will need to obtain some information, or documentation, from the person or organization providing the script. You will need to know the location of the script, whether it requires any specific names for the form controls, and whether it requires any hidden form elements.

A server-side script has been created at the author's website (<http://webdevbasics.net>) for students to use for this exercise. The documentation for the server-side script is listed below:

- Script URL: <http://webdevbasics.net/scripts/demo.php>
- Form method: `post`
- Script purpose: This script will accept form input and display the form control names and values in a web page. This is a sample script for student assignments. It demonstrates that server-side processing has been invoked. A script used by an actual website would perform a function such as sending an e-mail message or updating a database.

Now you will add the configuration required to use the `demo.php` server-side processing with a form. Launch a text editor and open `formcss.html` (see `chapter9/formcss.html` in the student files). Modify the `<form>` tag by adding an `action` attribute with a value of "<http://webdevbasics.net/scripts/demo.php>" and a `method` attribute with a value of "`post`". The HTML code for the revised `<form>` tag is

```
<form method="post" action="http://webdevbasics.net/scripts/demo.php">
```

Save your file as `contact.html` and test your web page in a browser. Your screen should look similar to Figure 9.19. Compare your work with the solution in the student files (`chapter9/9.5/contact.html`).

Now you are ready to test your form. You must be connected to the Internet to test your form successfully. Enter information in the form controls and click the submit button. You should see a confirmation page similar to the one shown in Figure 9.21.

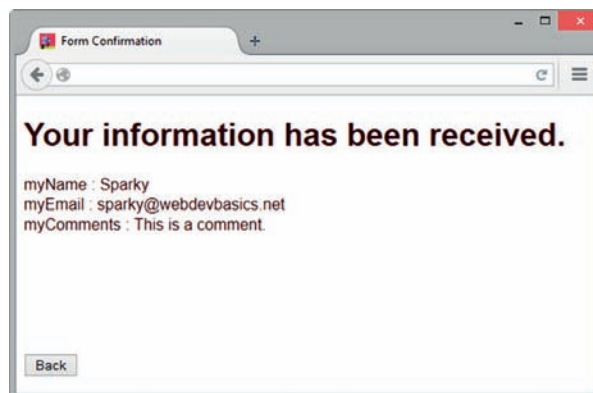


Figure 9.21 The server-side script has created this web page in response to the form

The `demo.php` script creates a web page that displays a message and the form information that you entered. Where did this confirmation page originate? This confirmation page was created by the server-side script on the `action` attribute in the form element. Sometimes students wonder what code is used in the `demo.php` file. Writing scripts for server-side processing is beyond the scope of this textbook. However, if you are curious, visit <http://webdevfoundations.net/8e/chapter9.html> to see the source code for this script.



FAQ What do I do if nothing happened when I tested my form?

Try these troubleshooting hints:

- Verify that your computer is connected to the Internet.
- Verify the spelling of the script location in the `action` attribute.
- Attention to detail is crucial!

Privacy and Forms

You've just learned how to collect information from your website visitors. Do you think that your visitors may want to know how you plan to use the information that you collect? The guidelines that you develop to protect the privacy of your visitors' information is called a **privacy policy**. Websites either indicate this policy on the form page itself or create a separate page that describes the privacy policy (and other company policies).

If you browse popular sites such as Amazon.com or eBay.com, you'll find links to their privacy policies (sometimes called a privacy notice) in the page footer area. The privacy policy of the Better Business Bureau can be found at <http://www.bbb.org/us/privacy-policy>. Include a privacy notice on your site to inform your visitors how you plan to use the information that they share with you. The Better Business Bureau (<https://www.bbb.org/dallas/for-businesses/bbb-sample-privacy-policy1/>) recommends that a privacy policy describes the type of information collected, the methods used to collect the information, the way that the information is used, the methods used to protect the information, and provisions for customers or visitors to control their personal information.

Server-Side Processing Resources

Sources of Free Remote-Hosted Form Processing

If your web host provider does not support server-side processing, free remotely hosted scripts may be an option. The script is not hosted on your server so you don't need to worry about installing it or whether your web host provider will support it. The disadvantage is that there may be some advertising displayed. The following are a few sites that offer this service:

- FormBuddy.com: <http://formbuddy.com>
- ExpressDB: <http://www.expressdb.com>
- FormMail: <http://www.formmail.com>
- Master.com: <http://www.master.com>

Sources of Free Server-Side Scripts

To use free scripts, you need to have access to a web server that supports the language used by the script. Contact your web host provider to determine what is supported. Be aware that many free web host providers do not support server-side processing (you get what you pay for!). Visit <http://scriptarchive.com> and <http://php.resourceindex.com> for free scripts and related resources.

Exploring Server-Side Processing Technologies

Many types of technologies can be used for server-side scripting, form processing, and information sharing:

- PHP: <http://www.php.net>
- Oracle JavaServer Pages Technology:
<http://www.oracle.com/technetwork/java/javaee/jsp>
- Adobe ColdFusion and Web Applications:
<http://www.adobe.com/products/coldfusion>
- Ruby on Rails: <http://www.rubyonrails.org>
- Microsoft .NET: <http://www.microsoft.com/net>
- Microsoft Active Server Pages: Active Server Pages:
<http://msdn.microsoft.com/en-us/library/ms972337.aspx>

Any of these technologies could be a good choice for future study. Web developers often learn the client side first (HTML, CSS, and JavaScript) and then progress to learning a server-side scripting or programming language.



Checkpoint 9.3

1. Describe server-side processing.
2. Why is communication needed between the developer of a server-side script and the web page designer?

9.9 HTML5 Form Controls

HTML5 offers a variety of new form controls for web developers that provide increased usability for web page visitors who are using modern browsers. For example, some new form controls offer built-in browser edits and validation. Future web designers will probably take these features for granted someday, but you are right in the middle of this huge advance in web page coding, so now is a great time to become familiar with the new form controls. The display and support of the new HTML5 form controls will vary by browser, but you can use them right now! Browsers that do not support the new input types will display them as text boxes and ignore unsupported attributes or elements. In this section, you'll explore the new HTML5 e-mail address, URL, telephone number, search field, datalist, slider, spinner, calendar, and color form controls.

E-mail Address Input

The **e-mail address input** form control is similar to the text box. Its purpose is to accept information that must be in e-mail format, such as "DrMorris2010@gmail.com". The input element with `type="email"` configures an e-mail address input form control.

Only browsers that support the HTML5 `email` attribute value will verify the format of the information. Other browsers will treat this form control as a text box. Attributes supported by the e-mail address input form control are listed in Table 9.2.

Figure 9.22 (see chapter9/email.html in the student files) shows an error message displayed by Firefox when text other than an e-mail address is entered. Note that the browser does not verify that the e-mail address actually exists, just that the text entered is in the correct format. The HTML is

```
<label for="myEmail">E-mail:</label>
<input type="email" name="myEmail" id="myEmail">
```

URL Input

The **URL input** form control is similar to the text box. It is intended to accept any valid type of URL or URI, such as “http://webdevfoundations.net”. The input element with `type="url"` configures a URL input form control. Only browsers that support the HTML5 `url` attribute value will verify the format of the information. Other browsers render this form control as a text box. Attributes supported by the URL input form control are listed in Table 9.2.

Figure 9.23 (see chapter9/url.html in the student files) shows an error message displayed by Firefox when text other than a URL is entered. Note that the browser does not verify that the URL actually exists, just that the text entered is in the correct format. The HTML is

```
<label for="myWebsite">Suggest a Website:</label>
<input type="url" name="myWebsite" id="myWebsite">
```

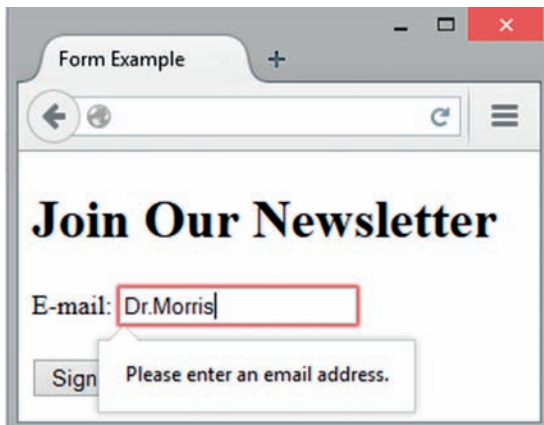


Figure 9.22 The browser displays an error message

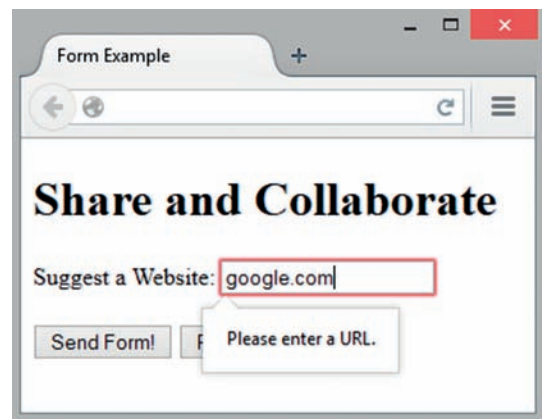


Figure 9.23 The browser displays an error message

Telephone Number Input

The **telephone number input** form control is similar to the text box. Its purpose is to accept a telephone number. The input element with `type="tel"` configures a telephone number input form control. An example is in the student files (chapter9/tel.html). Attributes supported by the telephone number input form control are listed in Table 9.2. Browsers that do not support `type="tel"` will render this form control as a text box. The HTML is

```
<label for="mobile">Mobile Number:</label>
<input type="tel" name="mobile" id="mobile">
```

Search Field Input

The **search field** is similar to the text box and is used to accept a search term. The input element with `type="search"` configures a search field input form control. An example is in the student files (chapter9/search.html). Attributes supported by the search field control are listed in Table 9.2. Browsers that do not support `type="search"` will render this form control as a text box. The HTML is

```
<label for="keyword">Search:</label>
<input type="search" name="keyword" id="keyword">
```



FAQ How can I tell which browsers support the new HTML5 form elements?

There's no substitute for testing. With that in mind, several resources are listed below that provide information about browser support for new HTML5 elements:

- When can I use . . . : <http://caniuse.com>
- HTML5 and CSS3 Support: <http://findmebyip.com/litmus>
- HTML5 and CSS3 Readiness: <http://html5readiness.com>
- The HTML5 Test: <http://html5test.com>
- HTML5 Web Forms and Browser Support: <http://www.standardista.com/html5>

Datalist Form Control

Figure 9.24 shows the **datalist form control** in action. Notice how a selection of choices is offered to the user along with a text box for entry. The datalist form control offers a convenient way to offer choices yet provide for flexibility on a form. The datalist is configured using three elements: an input element, the datalist element, and one or more option elements. Only browsers that support the HTML5 datalist element will display and process the datalist items. Other browsers ignore the datalist element and render the form control as a text box.

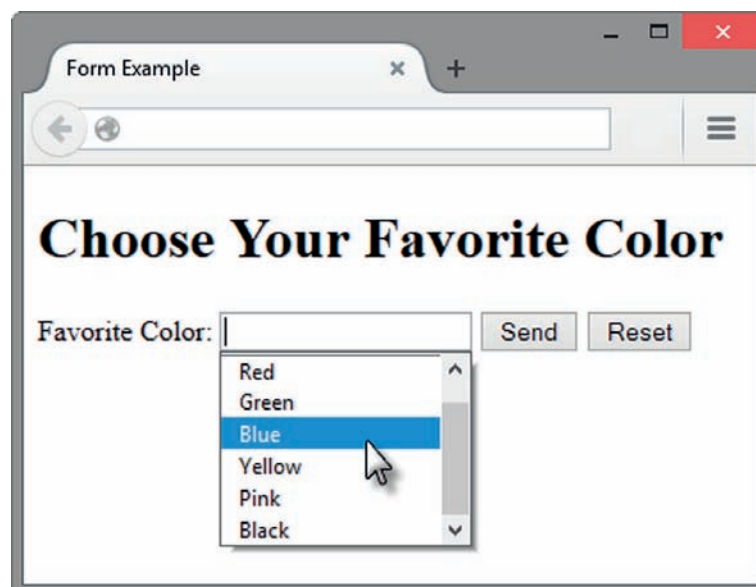


Figure 9.24 Firefox displays the datalist form control

The source code for the datalist is available in the student files (chapter9/list.html). The HTML is

```
<label for="color">Favorite Color:</label>
<input type="text" name="color" id="color" list="colors">
  <datalist id="colors">
    <option value="red"    label="Red">
    <option value="green"  label="Green">
    <option value="blue"   label="Blue">
    <option value="yellow" label="Yellow">
    <option value="pink"   label="Pink">
    <option value="black"  label="Black">
  </datalist>
```

Notice that the value of the **list** attribute on the input element is the same as the value of the **id** attribute on the datalist element. This creates the association between the text box and the datalist form control. One or more option elements can be used to offer predefined choices to your web page visitor. The option element's **label** attribute configures the text displayed in each list entry. The option element's **value** attribute configures the text sent to server-side processing when the form is submitted. The web page visitor can choose an option from the list (see Figure 9.24) or type directly in the text box, as shown in Figure 9.25.

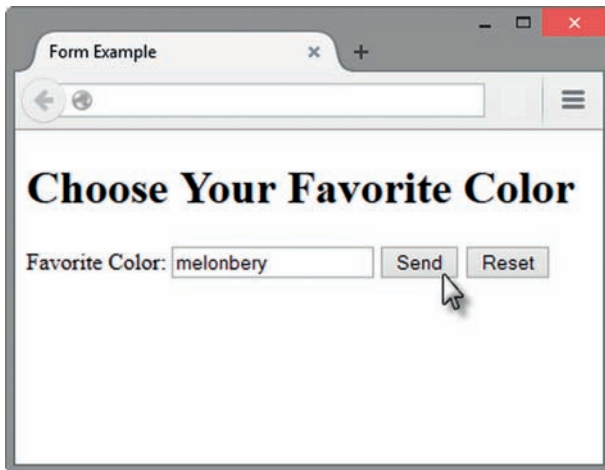


Figure 9.25 The list disappeared when the user began typing in the text box



FAQ Why should I learn about the new HTML5 form controls if they are not yet supported by all browsers?

The new form controls offer increased usability for your web page visitors who have modern browsers. And, they are backward compatible with older browsers, too. Browsers that do not support the new input types will display them as text boxes and ignore unsupported attributes or elements.

Slider Form Control

The **slider form control** provides a visual, interactive user interface that accepts numerical information. The input element with **type="range"** configures a slider control in which a number within a specified range is chosen. The default range is from 1 to 100. Only browsers that support the HTML5 **range** attribute value will display the interactive slider

control, shown in Figure 9.26 (see chapter9/range.html in the student files). Note the position of the slider in Figure 9.26; this resulted in the value 80 being chosen. The nondisplay of the value to the user may be a disadvantage of the slider control. Nonsupporting browsers render this form control as a text box, as shown in Figure 9.27.

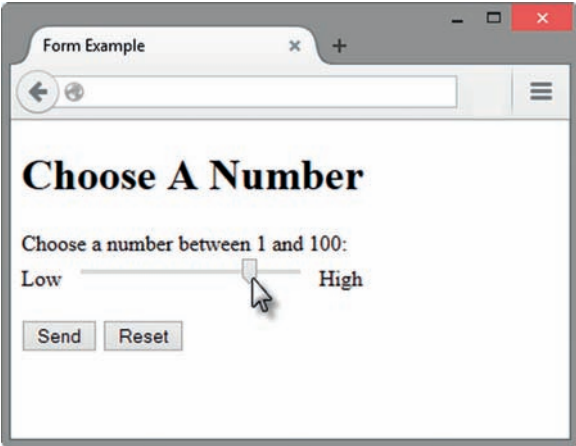


Figure 9.26 The Chrome browser displays the range form control. Screenshots of Internet Explorer. Copyright by Microsoft Corporation. Used by Permission of Microsoft Corporation

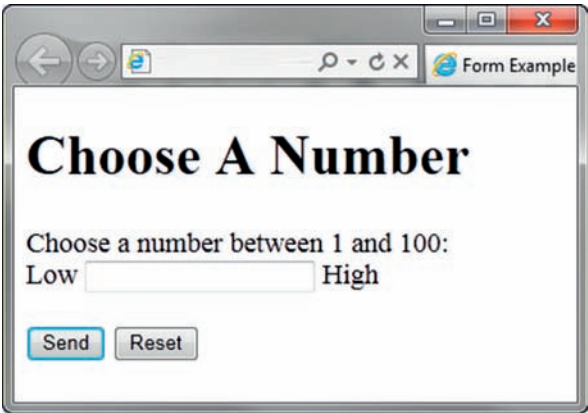


Figure 9.27 Internet Explorer 9 renders the range form control as a text box. Screenshots of Internet Explorer. Copyright by Microsoft Corporation. Used by Permission of Microsoft Corporation

The slider control accepts attributes listed in Tables 9.2 and 9.11. The `min`, `max`, and `step` attributes are new. Use the **min attribute** to configure the minimum range value. Use the **max attribute** to configure the maximum range value. The slider controls sets numeric values in increments, or steps, of 1. Use the **step attribute** to configure a value for the incremental steps between values to be other than 1.

The HTML for the slider control rendered in Figures 9.26 and 9.27 is

```
<label for="myChoice">Choose a number between 1 and 100:</label><br>
Low <input type="range" name="myChoice" id="myChoice" min="1"
max="100"> High
```

Table 9.11 Additional attributes for slider, spinner, and date/time form controls

| Attribute | Value | Purpose |
|-----------|--------------------------------|--|
| max | Maximum numeric value | HTML5 attribute for range, number, and date/time input controls; specifies a maximum value |
| min | Minimum numeric value | HTML5 attribute for range, number, and date/time input controls; specifies a minimum value |
| step | Incremental numeric step value | HTML5 attribute for range, number, and date/time input controls; specifies a value for incremental steps |

Spinner Form Control

The **spinner form control** displays an interface that accepts numerical information and provides feedback to the user. The input element with `type="number"` configures a spinner control in which the user can either type a number into the text box or select a number from a specified range. Only browsers that support the HTML5 `number` attribute value will display the interactive spinner control, shown in Figure 9.28 (see chapter9/spinner.html in the student files). Other browsers render this form control as a text box. You should expect increased support in the future.

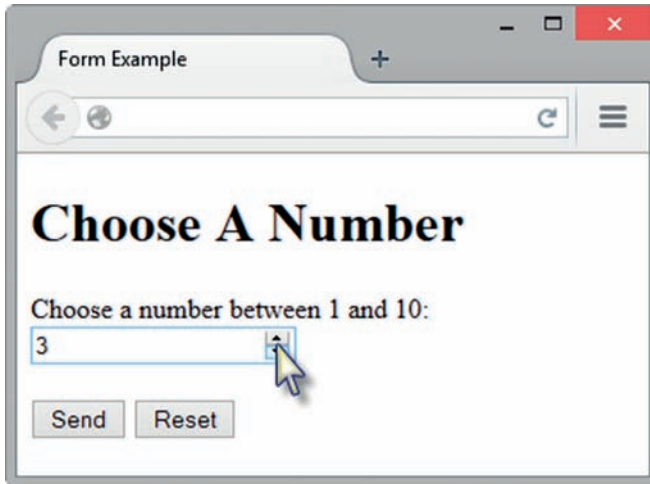


Figure 9.28 A spinner control displayed in the Google Chrome browser

The spinner control accepts attributes listed in Tables 9.2 and 9.11. Use the `min` attribute to configure the minimum value. Use the `max` attribute to configure the maximum value. The spinner control sets numeric values in increments, or steps, of 1. Use the `step` attribute to configure a value for the incremental step between values to be other than 1. The HTML for the spinner control displayed in Figure 9.28 is

```
<label for="myChoice">Choose a number between 1 and 10:</label>
<input type="number" name="myChoice" id="myChoice" min="1" max="10">
```

Calendar Form Control

HTML5 provides a variety of **calendar form controls** to accept date- and time-related information. Use the input element and configure the type attribute to specify a date or time control. Table 9.12 lists the HTML5 calendar date and time controls.

Table 9.12 Date and time controls

| Type | Attribute Value | Purpose | Format |
|----------------|-----------------|--|---|
| date | | A date | YYYY-MM-DD Example: January 2, 2016 is represented by "2016-01-02" |
| datetime | | A date and time with time zone information; note that the time zone is indicated by the offset from UTC time | YYYY-MM-DDTHH:MM:SS-##:##Z Example: January 2, 2016, at exactly 9:58 a.m. Chicago time (CST) is represented by "2016-01-02T09:58:00-06:00Z" |
| datetime-local | | A date and time without time zone information | YYYY-MM-DDTHH:MM:SS Example: January 2, 2016, at exactly 9:58 a.m. is represented by "2016-01-02T09:58:00" |
| time | | A time without time zone information | HH:MM:SS Example: 1:34 p.m. is represented by "13:34:00" |

(Continued)

Table 9.12 (Continued)

| Type | Attribute Value | Purpose | Format |
|-------|-----------------|------------------|---|
| month | | A year and month | YYYY-MM Example: January 2016 is represented by "2016-01" |
| week | | A year and week | YYYY-W##, where ## represents the week in the year Example: The third week in 2016 is represented by "2016-W03" |

The form in Figure 9.29 (see chapter9/date.html in the student files) uses the input element with `type="date"` to configure a calendar control with which the user can select a date. The HTML is

```
<label for="myDate">Choose a Date</label>
<input type="date" name="myDate" id="myDate">
```

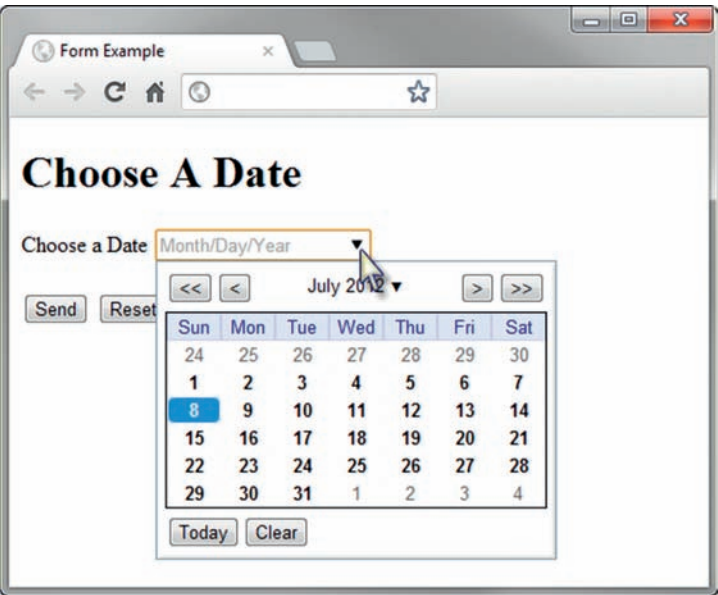


Figure 9.29 A date form control displayed in the Chrome browser

The date and time controls accept attributes listed in Tables 9.2 and 9.11. At the time this was written, only the Google Chrome, Microsoft Edge, and Opera browsers displayed a calendar interface for date and time controls. Other browsers currently render the **date and time form controls** as a text box, but you should expect increased support in the future.

Color-well Form Control

The **color-well form control** displays an interface that offers a color-picker interface to the user. The input element with `type="color"` configures a control with which the user can choose a color. Only browsers that support the HTML5 `color` attribute value will display a color-picker interface, shown in Figure 9.30 (see chapter9/color.html in the student files). Other browsers render this form control as a text box.

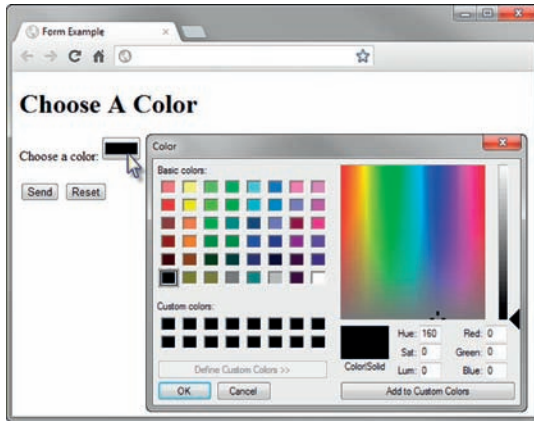


Figure 9.30 The Chrome browser supports the color-well form control

The HTML for the color-well form control rendered in Figure 9.30 is

```
<label for="myColor">Choose a color:</label>
<input type="color" name="myColor" id="myColor">
```

In the next Hands-On Practice, you'll get some experience with the new HTML5 form controls.



Hands-On Practice 9.6

In this Hands-On Practice, you will code HTML5 form controls as you configure a form that accepts a name, e-mail address, rating value, and comments from a website visitor. Figure 9.31 displays the form in the Google Chrome browser, which supports the HTML5 features used in the Hands-On Practice. Figure 9.32 displays the form in Internet Explorer 9, which does not support the HTML5 features. Notice that the form is enhanced in Google Chrome, but is still usable in both browsers, demonstrating the concept of progressive enhancement.

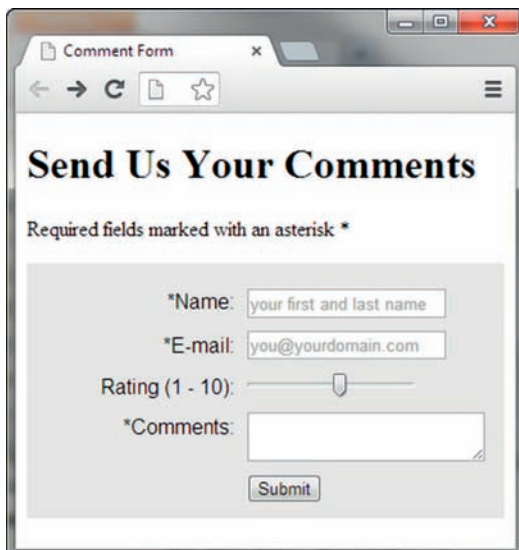


Figure 9.31 The form displayed in the Chrome browser

The screenshot shows a web browser window with the title 'Comment Form'. The main heading is 'Send Us Your Comments'. Below the heading is a note: 'Required fields marked with an asterisk *'. The form contains four fields: '*Name:' with a text input, '*E-mail:' with a text input, 'Rating (1 - 10):' with a text input, and '*Comments:' with a text area. A 'Submit' button is located at the bottom of the form.

Figure 9.32 The form displayed in the Internet Explorer 9 browser. Screenshots of Internet Explorer. Copyright by Microsoft Corporation. Used by Permission of Microsoft Corporation.

To get started, launch a text editor and open `chapter9/formcss.html` in the student files, shown in Figure 9.19. Save the file as `form6.html`. You will modify the file to create a web page similar to the examples in Figures 9.31 and 9.32.

1. Modify the title element to display the text "Comment Form". Configure the text contained within the `h1` element to be "Send Us Your Comments". Add a paragraph to indicate "Required fields are marked with an asterisk *."
2. Modify the embedded styles:
 - a. Configure the form element selector to have a minimum width of 400 pixels.
 - b. Change the width of the label element selector to 150 pixels.
 - c. Change the left margin of `#mySubmit` to 160 pixels.
3. Modify the form element to submit the form information, using the `post` method, to the form processor at `http://webdevbasics.net/scripts/demo.php`:

```
<form method="post" action="http://webdevbasics.net/scripts/demo.php">
```

4. Modify the form controls.
 - a. Configure the name, e-mail, and comment information to be required. Use an asterisk to inform your web page visitor about the required fields.
 - b. Code `type="email"` instead of `type="input"` for the e-mail address.
 - c. Use the `placeholder` attribute (refer to Table 9.2) to provide hints to the user in the name and e-mail form controls.
5. Add a slider control (use `type="range"`) to generate a value from 1 to 10 for the rating.

The HTML for the form follows:

```
<form method="post"
action="http://webdevbasics.net/scripts/demo.php">
  <label for="myName">*Name:</label>
  <input type="text" name="myName" id="myName"
    required="required" placeholder="your first and last name">
  <label for="myEmail">*E-mail:</label>
  <input type="email" name="myEmail" id="myEmail"
    required="required" placeholder="you@yourdomain.com">
```

```

<label for="myRating">Rating (1 - 10):</label>
<input type="range" name="myRating" id="myRating" min="1" max="10">
<label for="myComments">*Comments:</label>
<textarea name="myComments" id="myComments" rows="2" cols="20"
    required="required"></textarea>
<input id="mySubmit" type="submit" value="Submit">
</form>

```

6. Save form6.html and test your web page in a browser. If you use a browser that supports the HTML5 features used in the form (such as Google Chrome), your page should look similar to Figure 9.31. If you use a browser that does not support the form's HTML5 attributes (such as Internet Explorer 8), your form should look similar to Figure 9.32. The display in other browsers will depend on the level of HTML5 support. See HTML5 Web Forms and Browser Support (<http://www.standardista.com/html5/html5-web-forms>) for an HTML5 browser support list.
7. Try submitting the form without entering any information. Figure 9.33 shows the result when using Firefox. Note the error message which indicates that the name field is required. Compare your work with the solution in the student files (chapter9/9.6/form.html).

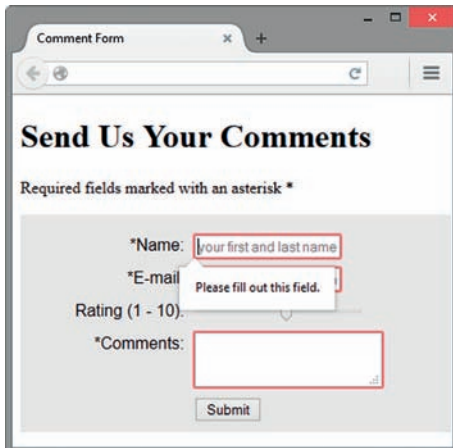


Figure 9.33 The Firefox browser displays an error message

As this Hands-On Practice demonstrated, support of the new HTML5 form control attributes and values is not uniform. It will be some time before all browsers support these new features. Design forms with progressive enhancement in mind and be aware of both the benefits and the limitations of using the new HTML5 features.

HTML5 and Progressive Enhancement

Use HTML5 form elements with the concept of progressive enhancement in mind. Nonsupporting browsers will display text boxes in place of form elements that are not recognized. Supporting browsers will display and process the new form controls. This is progressive enhancement in action: Everyone sees a usable form and visitors who are using modern browsers benefit from the enhanced features.

Chapter Summary

This chapter introduced the use of forms on web pages. You learned how to configure form controls, provide for accessibility, and configure a form to access server-side processing. You also explored new HTML5 form elements. Visit the textbook website at <http://www.webdevfoundations.net> for examples, the links listed in this chapter, and updated information.

Key Terms

| | | |
|--------------------------------|----------------------|------------------------|
| <code><button></code> | e-mail address input | radio button |
| <code><fieldset></code> | fieldset element | required attribute |
| <code><form></code> | for attribute | reset button |
| <code><input></code> | form | scrolling text box |
| <code><label></code> | form controls | search field |
| <code><legend></code> | form element | select element |
| <code><option></code> | hidden input control | select list |
| <code><select></code> | image button | server-side scripting |
| <code><textarea></code> | input element | slider form control |
| accesskey attribute | label element | spinner form control |
| action attribute | legend element | step attribute |
| button element | list attribute | submit button |
| calendar form controls | max attribute | tabindex attribute |
| check box | method attribute | telephone number input |
| color-well form control | min attribute | text box |
| Common Gateway Interface (CGI) | name attribute | textarea element |
| datalist form control | option element | URL input |
| date and time form controls | password box | value attribute |
| direct execution | privacy policy | |

Review Questions

Multiple Choice

1. You would like to conduct a survey and ask your web page visitors to vote for their favorite search engine. Which of the following form controls is best to use for this purpose?
 - a. check box
 - b. radio button
 - c. text box
 - d. scrolling text box
2. You would like to conduct a survey and ask your web page visitors to indicate the web browsers that they use. Which of the following form controls is best to use for this purpose?
 - a. check box
 - b. radio button
 - c. text box
 - d. scrolling text box
3. Forms contain various types of _____, such as text boxes and buttons, which accept information from a web page visitor.
 - a. hidden elements
 - b. labels
 - c. form controls
 - d. legends
4. Choose the HTML tag that would configure a text box with the name "city" and a width of 40 characters.
 - a. `<input type="text" id="city" width="40">`
 - b. `<input type="text" name="city" size="40">`
 - c. `<input type="text" name="city" space="40">`
 - d. `<input type="text" width="40">`

5. Which of the following form controls would be appropriate for an area that your visitors can use to type in their e-mail address?
 - a. select list
 - b. text box
 - c. scrolling text box
 - d. label
6. Which of the following form controls would be appropriate for an area that your visitors can use to type in comments about your website?
 - a. text box
 - b. select list
 - c. radio button
 - d. scrolling text box
7. Which attribute of the form element is used to specify the name and location of the script that will process the form field values?
 - a. action
 - b. process
 - c. method
 - d. id
8. Which HTML tag would configure a scrolling text box with the name "comments", 2 rows, and 30 characters?
 - a. `<textarea name="comments" width="30" rows="2"></textarea>`
 - b. `<input type="textarea" size="30" name="comments" rows="2">`
 - c. `<textarea name="comments" rows="2" cols="30"></textarea>`
 - d. `<input type="comments" rows="2" name="comments" cols="30">`
9. You would like to accept a number that's in a range from 1 to 50. The user needs visual verification of the number they selected. Which of the following form controls is best to use for this purpose?
 - a. spinner
 - b. check box
 - c. radio button
 - d. slider
10. Choose the HTML that would associate a label displaying the text "E-mail:" with the e-mail text box.
 - a. E-mail `<input type="textbox" name="email" id="email">`
 - b. `<label>E-mail: </label><input type="text" name="email" id="email">`
 - c. `<label for="email">E-mail: </label><input type="text" name="email" id="emailaddress">`
 - d. `<label for="email">E-mail: </label><input type="text" name="email" id="email">`
11. What will happen when a browser encounters a new HTML5 form control that it does not support?
 - a. The computer will shut down.
 - b. The browser will crash.
 - c. The browser will display an error message.
 - d. The browser will display an input text box.

Fill in the Blank

12. To limit the number of characters that a text box will accept, use the _____ attribute.
13. To group a number of form controls visually on the page, use the _____ element.
14. To cause a number of radio buttons to be treated as a single group, the value of the _____ attribute must be identical.

Short Answer

15. Describe at least three form controls that could be used to allow a visitor to your web page to select a color.

Apply Your Knowledge

1. **Predict the Result.** Draw and write a brief description of the web page that will be created with the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Predict the Result</title>
<meta charset="utf-8">
</head>
```

```

<body>
<h1>Contact Us</h1>
<form action="myscript.php">
<fieldset><legend>Complete the form and a consultant will contact
you.</legend>
E-mail: <input type="text" name="email" id="email" size="40">
<br>Please indicate which services you are interested in:<br>
<select name="inquiry" id="inquiry" size="1">
    <option value="development">Web Development</option>
    <option value="redesign">Web Redesign</option>
    <option value="maintain">Web Maintenance</option>
    <option value="info">General Information</option>
</select>
<br>
<input type="submit">
</fieldset>
</form>
<nav><a href="index.html">Home</a>
<a href="services.html">Services</a>
<a href="contact.html">Contact</a></nav>
</body>
</html>

```

- 2. Fill in the Missing Code.** This web page configures a survey form to collect information on the favorite search engine used by web page visitors. The form action should submit the form to the server-side script, called survey.php. Some HTML tags and their attributes, indicated by `<_>`, are missing. Some HTML attribute values, indicated by `"_"`, are missing.

```

<!DOCTYPE html>
<html lang="en">
<head>
<title>Fill in the Missing Code</title>
<meta charset="utf-8">
</head>
<body>
<h1>Vote for your favorite Search Engine</h1>
<form method="_" action="_">
    <input type="radio" name="_" id="Ysurvey" value="Yahoo">
Yahoo!<br>
    <input type="radio" name="survey" id="Gsurvey" value="Google">
Google<br>
    <input type="radio" name="_" id="Bsurvey" value="Bing"> Bing<br>
    <_>
</form>
</body>
</html>

```

- 3. Find the Error.** Find the coding errors in the following subscription form:

```

<!DOCTYPE html>
<html lang="en">

```

```
<head>
<title>Find the Error</title>
<meta charset="utf-8">
</head>
<body>
<p>Subscribe to our monthly newsletter and receive free coupons!</p>
<form action="get" method="newsletter.php">
  <label>E-mail: <input type="text" name="email" id="email"
    char="40"></label>
  <br>
  <input button="submit"> <input type="reset">
</form>
</body>
</html>
```

Hands-On Exercises

1. Write the HTML code to create the following:
 - a. A text box named user that will be used to accept the user name of web page visitors. The text box should allow a maximum of 30 characters to be entered.
 - b. A group of radio buttons that website visitors can check to vote for their favorite month of the year.
 - c. A select list that asks website visitors to select their favorite social networking site
 - d. A fieldset and legend with the text “Shipping Address” around the following form controls:
AddressLine1, AddressLine2, City, State, ZIP
 - e. An image called signup.gif as an image button on a form
 - f. A hidden input control with the name userid
 - g. A password box form control with the name pword
 - h. A form tag to invoke server-side processing using
<http://webdevbasics.net/scripts/demo.php> and the post method
2. Write the HTML to create a form that accepts requests for a brochure to be sent in the mail. Sketch out the form on paper before you begin.
3. Create a web page with a form that accepts feedback from website visitors. Use the HTML5 input `type="email"` along with the `required` attribute to configure the browser to verify the data entered. Also configure the browser to require user comments with a maximum length of 1600 characters accepted. Place your name and e-mail address at the bottom of the page. *Hint:* Sketch out the form on paper before you begin.
4. Create a web page with a form that accepts a website visitor’s name, e-mail, and birthdate. Use the HTML5 `type="date"` attribute to configure a calendar control on browsers that support the attribute value. Place your name and e-mail address at the bottom of the page. *Hint:* Sketch out the form on paper before you begin.
5. Write a web page that contains a music survey form similar to the example shown in Figure 9.34.

Figure 9.34
Sample music
survey form

Music Survey

Name:

E-Mail:

Select Your Favorite Types of Music:

☐ Pop
☐ Classical
☐ Rock
☐ Folk
☐ Rap
☐ Other

Select how often you purchase Music CDs:

☐ Weekly
☒ A few CDs each year
☐ Monthly
☐ Never purchase

Select the locations you listen to Music:

At home
In the car
Anywhere (MP3 Player)

What role does music play in your life?

Include the following form controls:

- Text box for name
- E-mail address input form control for the e-mail address
- A scrolling text box that is 60 characters wide and 3 rows high
- A radio button group with at least three choices
- A check box group with at least three choices
- A select box that initially shows three items but contains at least four items
- A submit button
- A reset button
- Use the fieldset and legend elements as shown in Figure 9.34 to configure the display of form areas with radio buttons and checkboxes.

Use a CSS to configure the display of your form. Place your name and e-mail address at the bottom of the page.

Web Research

1. This chapter mentioned a number of sources of free remotely hosted scripts, including FormBuddy.com (<http://formbuddy.com>), FormMail (<http://www.formmail.com>), ExpressDB (<http://www.expressdb.com>), and Master.com (<http://master.com>). Visit two of these sites or use a search engine to find other resources for free remotely hosted

scripts. Register (if necessary) and examine the website to see exactly what is offered. Most sites that provide remotely hosted scripts have a demo you can view or try. If you have time (or your instructor asks you to), follow the directions and access a remotely hosted script from one of your web pages. Now that you've at least been through a demo of the product or tried it yourself (even better!), it's time to write your review.

Create a web page that lists the two resource sites you chose and provides a comparison of what they offer. List the following for each website:

- Ease of registration
- Number of scripts or services offered
- Types of scripts or services offered
- Site banner or advertisement
- Ease of use
- Your recommendation

Provide links to the resource sites you reviewed and place your name and e-mail address at the bottom of the page.

2. Search the Web for a web page that uses an HTML form. Print the browser view of the page. Print out the source code of the web page. Using the printout, highlight or circle the tags related to forms. On a separate sheet of paper, create some HTML notes by listing the tags and attributes related to the forms found on your sample page along with a brief description of their purpose.
3. Choose one server-side technology mentioned in this chapter such as PHP, JSP, Ruby on Rails, or ASP.NET. Use the resources listed in the chapter as a starting point, but also search the Web for additional resources on the server-side technology you have chosen. Create a web page that lists at least five useful resources along with information about each that provides the name of the site, the URL, a brief description of what is offered, and a recommended page (such as a tutorial, free script, and so on). Place your name in an e-mail link on the web page.

Focus on Web Design

The design of a form, such as the justification of the labels, the use of background colors, and even the order of the form elements can either increase or decrease the usability of a form. Visit some of the following resources to explore form design:

- Web Application Form Design: http://www.uie.com/articles/web_forms
- 7 Common Web Form Design Mistakes to Avoid: <http://www.formassembly.com/blog/web-form-design>
- 10 Tips to a Better Form: <http://particletree.com/features/10-tips-to-a-better-form>
- Sensible Forms: <http://www.alistapart.com/articles/sensibleforms>
- Best Practices for Form Design: http://static.lukew.com/webforms_lukew.pdf

Create a web page that lists the URLs of at least two useful resources along with a brief description of the information you found most interesting or valuable. Design a form on the web page that applies what you've just learned in your exploration of form design. Place your name in an e-mail link on the web page.



WEBSITE CASE STUDY

Adding a Form

Each of the following case studies continues throughout most of the textbook. This chapter adds a page containing a form that invokes server-side processing to the websites.

JavaJam Coffee House

See Chapter 2 for an introduction to the JavaJam Coffee House case study. Figure 2.30 shows a site map for the JavaJam site. Use the Chapter 8 JavaJam website as a starting point for this case study. You will create the new Jobs page that contains a form. You have four tasks in this case study:

1. Create a new folder for this JavaJam case study.
2. Modify the style sheet (javajam.css) to configure style rules for the new form.
3. Create the new Jobs page shown in Figure 9.35.
4. Configure HTML5 form controls.



Figure 9.35 JavaJam Jobs page

Hands-On Practice Case Study

Task 1: Create a Folder. Create a folder called javajam9. Copy all of the files from your Chapter 8 javajam8 folder into the javajam9 folder.

Task 2: Configure the CSS. Modify the external style sheet (javajam.css). Open javajam.css in a text editor. Review Figure 9.35 and the wireframe in Figure 9.36. Notice how the text labels for the form controls are on the left side of the content area but contain right-aligned text. Notice the empty vertical space between each form control. Configure CSS as indicated below:

1. Create a form element selector with a style declaration that sets 2em of padding.
2. Configure a label element selector to float to the left with block display. Set the text alignment to right, assign a width of 8em, and set 1em of right padding.

3. Configure the input element and textarea element selectors with block display and 1em of bottom margin.
4. Configure a selector for an id named `mySubmit` that sets the left margin to 9.5em.

Save the `javajam.css` file.

Task 3: Create the Jobs Page. Use the Menu page as the starting point for the Jobs page. Launch a text editor and open `menu.html`. Save the file as `jobs.html`. Modify your `jobs.html` file to look similar to the Jobs page (shown in Figure 9.35) as follows:

1. Change the page title to an appropriate phrase.
2. The Jobs page will contain an `h2`, a paragraph, and a form in the main element. Delete the `div` within the main element. Delete the table in the main element.
3. Edit the text within the `h2` element to say "Jobs at JavaJam". Replace the text in the paragraph with the following: "Want to work at JavaJam? Fill out the form below to start your application."
4. Prepare to code the HTML for the form area. Begin with a form element that uses the post method and the action attribute to invoke server-side processing. Unless directed otherwise by your instructor, configure the action attribute to send the form data to `http://webdevbasics.net/scripts/javajam8.php`.
5. Configure the form control for the Name information. Create a label element that contains the text "Name:". Create a text box named `myName`. Use the `for` attribute to associate the label element with the form control.
6. Configure the form control for the E-mail information. Create a label element that contains the text "E-mail:". Create a text box named `myEmail`. Use the `for` attribute to associate the label element with the form control.
7. Configure the Experience area on the form. Create a label element that contains the text "Experience:". Create a textarea element named `myExperience` with `rows` set to 2 and `cols` set to 20. Use the `for` attribute to associate the label element with the form control.
8. Configure the submit button. Code an input element with `type="submit"` and `value="Apply Now"`. Assign the input element to an id named `mySubmit`.
9. Code an ending `</form>` tag on a blank line after the submit button.

Save your file and test your web page in a browser. It should look similar to the page shown in Figure 9.35. If you are connected to the Internet, submit the form. This will send your form information to the server-side script configured in the form tag. A confirmation page that lists the form information and their corresponding names will be displayed.

Task 4: Configure HTML5 Form Controls. Get more practice with the new HTML5 elements by modifying the form on the Jobs page to use HTML5 attributes and values. Modify the `jobs.html` file in a text editor.

1. Add the following sentence to the paragraph above the form: "Required fields are marked with an asterisk (*)."
2. Use the `required` attribute to require the name, e-mail, and experience form controls to be entered. Add an asterisk at the beginning of each label text.
3. Configure the input element for the e-mail address with `type="email"`.

Save your file and display your web page in a browser. Submit the form with missing information or only a partial e-mail address. Depending on the browser's level of HTML5 support,

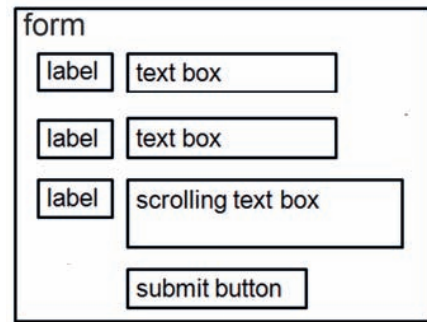


Figure 9.36 Wireframe for the form



Figure 9.37 The Jobs page with HTML5 form controls

the browser may perform form validation and display an error message. Figure 9.37 shows the Jobs page rendered in the Firefox browser with an incorrectly formatted e-mail address.

This task provided you with additional practice using the new HTML5 attributes and values. The display and functioning of browsers will depend on the level of HTML5 support. See HTML5 Web Forms and Browser Support (<http://www.standardista.com/html5/html5-web-forms>) for an HTML5 browser support list.

Fish Creek Animal Hospital

See Chapter 2 for an introduction to the Fish Creek Animal Hospital case study. Figure 2.34 shows a site map for Fish Creek. Use the Chapter 8 Fish Creek website as a starting point for this case study. You will create the new Contact page that contains a form. You have four tasks in this case study:

1. Create a new folder for this Fish Creek case study.
2. Modify the style sheet (fishcreek.css) to configure style rules for the new form.
3. Create the new Contact page shown in Figure 9.38.
4. Configure HTML5 form controls.

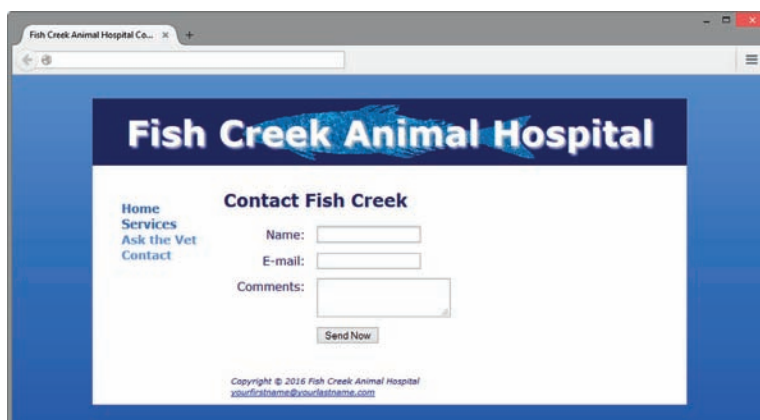


Figure 9.38 Fish Creek Contact page

Hands-On Practice Case Study

Task 1: Create a Folder. Create a folder called fishcreek9. Copy all of the files from your Chapter 8 fishcreek8 folder into the fishcreek9 folder.

Task 2: Configure the CSS. Modify the external style sheet (fishcreek.css). Open fishcreek.css in a text editor. Review Figure 9.38 and the wireframe in Figure 9.39. Notice how the text labels for the form controls are on the left side of the content area but contain right-aligned text. Notice the empty vertical space between each form control. Configure CSS as indicated below:

1. Create a label element selector to float to the left with block display. Set the text alignment to right, assign a width of 8em, and set 1em right padding.
2. Configure the input element and textarea element selectors with block display and 1em of bottom margin.
3. Configure an id named `mySubmit` with a 9.5em left margin.

Save the fishcreek.css file.

Task 3: Create the Contact Page. Use the Ask the Vet page as the starting point for the Contact page. Launch a text editor and open askvet.html. Save the file as contact.html. Modify your contact.html file to look similar to the Contact page (shown in Figure 9.38) as follows:

1. Change the page title to an appropriate phrase.
2. The Contact page will display a form in the main element. Delete the paragraph and description list in the main element.
3. Add an h2 element that contains the following text: “Contact Fish Creek”.
4. Prepare to code the HTML for the form area. Begin with a form element that uses the post method and the action attribute to invoke server-side processing. Unless directed otherwise by your instructor, configure the action attribute to send the form data to `http://webdevbasics.net/scripts/fishcreek.php`.
5. Configure the form control for the Name information. Create a label element that contains the text “Name:”. Create a text box named `myName`. Use the `for` attribute to associate the label element with the form control.
6. Configure the form control for the E-mail information. Create a label element that contains the text “E-mail:”. Create a text box named `myEmail`. Use the `for` attribute to associate the label element with the form control.
7. Configure the Comments area on the form. Create a label element that contains the text “Comments:”. Create a textarea element named `myComments` with `rows` set to 2 and `cols` set to 20. Use the `for` attribute to associate the label element with the form control.
8. Configure the submit button on the form. Configure “Send Now” to display on the button. Assign the input element to the id named `mySubmit`.
9. Code an ending `</form>` tag on a blank line after the submit button.

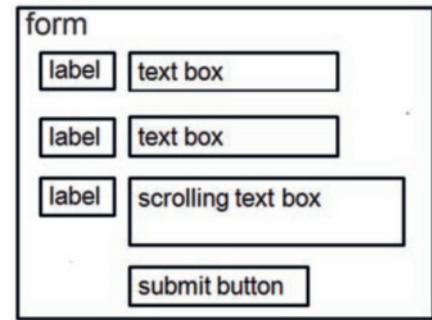


Figure 9.39 Wireframe for the form

Save your file and test your web page in a browser. It should look similar to the page shown in Figure 9.38. If you are connected to the Internet, submit the form. This will send your form information to the server-side script configured in the form tag. A confirmation page that lists the form information and their corresponding names will be displayed.

Task 4: Configure HTML5 Form Controls. Get more practice with the new HTML5 elements by modifying the form on the Contact page to use HTML5 attributes and values. Modify the contact.html file in a text editor.

1. Add a paragraph above the form with the following sentence: “Required fields are marked with an asterisk (*).”
2. Use the `required` attribute to require the name, e-mail, and comments form controls to be entered. Add an asterisk at the beginning of each label text.
3. Configure the input element for the e-mail address with `type="email"`.

Save your file and display your web page in a browser. Submit the form with missing information or only a partial e-mail address. Depending on the browser’s level of HTML5 support, the browser may perform form validation and display an error message. Figure 9.40 shows the Contact page rendered in the Firefox with missing required information.

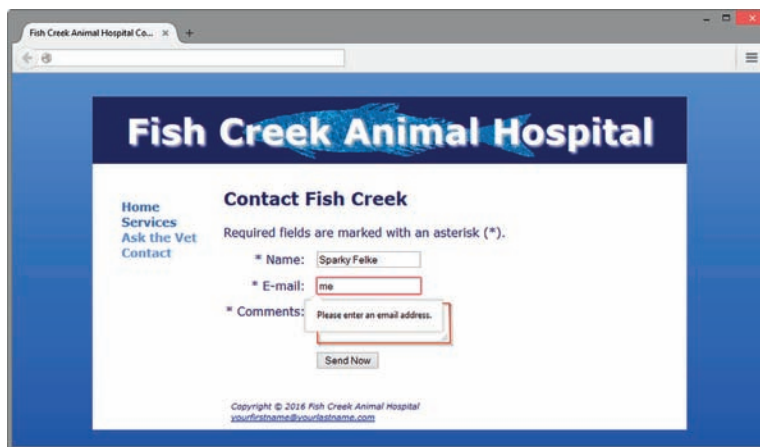


Figure 9.40 The Contact page with HTML5 form controls

This task provided you with additional practice using the new HTML5 attributes and values. The display and functioning of browsers will depend on the level of HTML5 support. See HTML5 Web Forms and Browser Support (<http://www.standardista.com/html5/html5-web-forms>) for an HTML5 browser support list.

Pacific Trails Resort

See Chapter 2 for an introduction to the Pacific Trails Resort case study. Figure 2.38 shows a site map for Pacific Trails. Use the Chapter 8 Pacific Trails website as a starting point for this case study. You will create the new Reservations page that contains a form. You have four tasks in this case study:

1. Create a new folder for this Pacific Trails case study.
2. Modify the style sheet (pacific.css) to configure style rules for the new form.

3. Create the new Reservations page shown in Figure 9.41.
4. Configure HTML5 form controls.

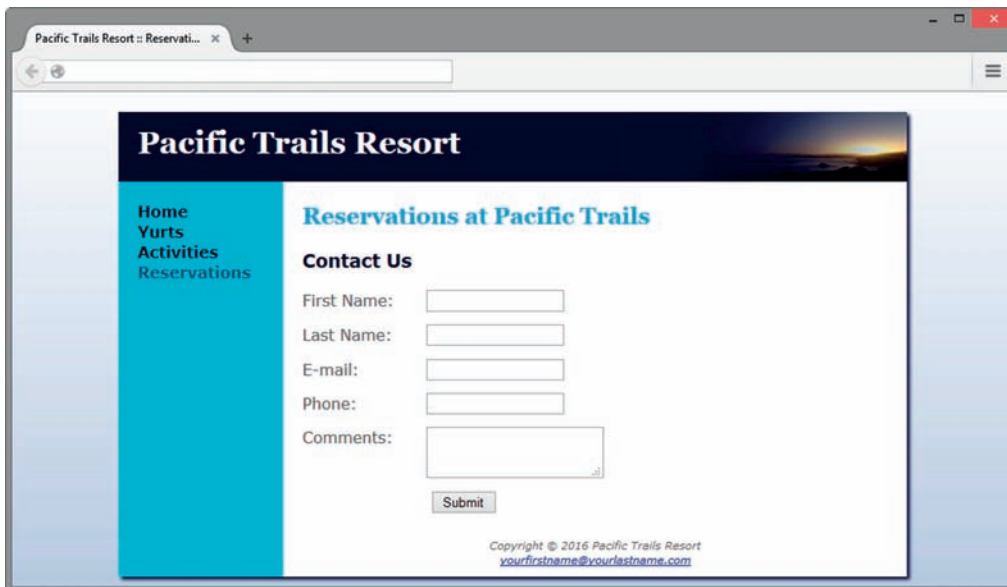


Figure 9.41 Pacific Trails Reservations page

Hands-On Practice Case Study

Task 1: Create a Folder. Create a folder called `pacific9`. Copy all of the files from your Chapter 8 `pacific8` folder into the `pacific9` folder.

Task 2: Configure the CSS. Modify the external style sheet (`pacific.css`). Open `pacific.css` in a text editor. Review Figure 9.41 and the wireframe in Figure 9.42. Notice how the text labels for the form controls are on the left side of the content area. Notice the empty vertical space between each form control. Configure CSS as indicated below:

1. Create a label element selector to float to the left with block display. Set the width to 8em. Configure 1em right padding.
2. Configure the input element and textarea element selectors with block display and 1em of bottom margin.
3. Configure an id named `mySubmit` with a 10em left margin.

Save the `pacific.css` file.

Task 3: Create the Reservations Page. Use the Home page as the starting point for the Reservations page. Launch a text editor and open `index.html`. Save the file as `reservations.html`. Modify your `reservations.html` file to look similar to the Reservations page (shown in Figure 9.41) as follows:

1. Change the page title to an appropriate phrase.
2. Delete all HTML tags and content within the main element except for the `h2` element and text.

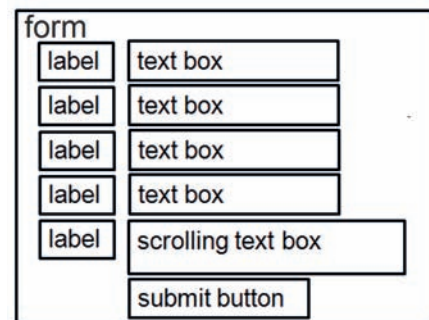


Figure 9.42 Wireframe for the form

3. Replace the text contained within the `<h2>` tags with: “Reservations at Pacific Trails”.
4. Configure an `h3` element on a line under the `h2` with the following text: “Contact Us”.
5. Prepare to code the HTML for the form area. Begin with a form element that uses the `post` method and the `action` attribute to invoke server-side processing. Unless directed otherwise by your instructor, configure the `action` attribute to send the form data to `http://webdevbasics.net/scripts/pacific.php`.
6. Configure the form control for the First Name information. Create a label element that contains the text “First Name:”. Create a text box named `myFName`. Use the `for` attribute to associate the label element with the form control.
7. Configure the form control for the Last Name information. Create a label element that contains the text “Last Name:”. Create a text box named `myLName`. Use the `for` attribute to associate the label element with the form control.
8. Configure the form control for the E-mail information. Create a label element that contains the text “E-mail:”. Create a text box named `myEmail`. Use the `for` attribute to associate the label element with the form control.
9. Configure the form control for the Phone information. Create a label element that contains the text “Phone:”. Create a text box named `myPhone`. Use the `for` attribute to associate the label element with the form control.
10. Configure the Comments area on the form. Create a label element that contains the text “Comments:”. Create a `textarea` element named `myComments` with `rows` set to 2 and `cols` set to 20. Use the `for` attribute to associate the label element with the form control.
11. Configure the submit button on the form. Configure “Submit” to display on the button. Assign the `input` element to the `id` named `mySubmit`.
12. Code an ending `</form>` tag on a blank line after the submit button.

Save your file and test your web page in a browser. It should look similar to the page shown in Figure 9.41. If you are connected to the Internet, submit the form. This will send your form information to the server-side script configured in the `form` tag. A confirmation page that lists the form information and their corresponding names will be displayed.

Task 4: Configure HTML5 Form Controls. Get more practice with the new HTML5 elements by modifying the form on the Reservations page to use HTML5 attributes and values. Modify the `reservations.html` file in a text editor.

1. Add a paragraph above the form with the following sentence: “Required information is marked with an asterisk (*).”
2. Use the `required` attribute to require the first name, last name, e-mail, and comments form controls to be entered. Add an asterisk at the beginning of each label text.
3. Configure the input element for the e-mail address with `type="email"`.
4. Configure the input element for the phone number with `type="tel"`.
5. Add a calendar form control to process a reservation request date (use `type="date"`).
6. Add a spinner form control to process a value between 1 and 14 to indicate the number of nights for the length of stay (use `type="number"`). Use the `min` and `max` attributes to configure the range of values.

Save your file and display your web page in a browser. Submit the form with missing information or only a partial e-mail address. Depending on the browser’s level of HTML5 support, the browser may perform form validation and display an error message. Figure 9.43 shows the Reservations page rendered in the Google Chrome with an incorrectly formatted e-mail address.

Pacific Trails Resort

Home
Yurts
Activities
Reservations

Reservations at Pacific Trails

Contact Us

Required information is marked with an asterisk (*).

* First Name:

* Last Name:

* E-mail:

Phone: Please fill out this field.

Arrival Date:

Nights:

* Comments:

Copyright © 2016 Pacific Trails Resort
yourfirstname@yourlastname.com

Figure 9.43 The Reservations page with HTML5 form controls

This task provided you with additional practice using the new HTML5 attributes and values. The display and functioning of browsers will depend on the level of HTML5 support. See HTML5 Web Forms and Browser Support (<http://www.standardista.com/html5/html5-web-forms>) for an HTML5 browser support list.

Path of Light Yoga Studio

See Chapter 2 for an introduction to the Path of Light Yoga Studio case study. Figure 2.42 shows a site map for Path of Light Yoga Studio. Use the Chapter 8 Path of Light Yoga Studio website as a starting point for this case study. You will create the new Contact page that uses a form. You have four tasks in this case study:

1. Create a new folder for this Path of Light Yoga Studio case study.
2. Modify the style sheet (yoga.css) to configure style rules for the new form.
3. Create the Contact page shown in Figure 9.44.
4. Configure HTML5 form controls.

Path of Light Yoga Studio

Home
Classes
Schedule
Contact

Contact Path of Light Yoga Studio

Name:

E-mail:

Comments:

Copyright © 2016 Path of Light Yoga
yourfirstname@yourlastname.com

Figure 9.44 Path of Light Yoga Studio Contact page

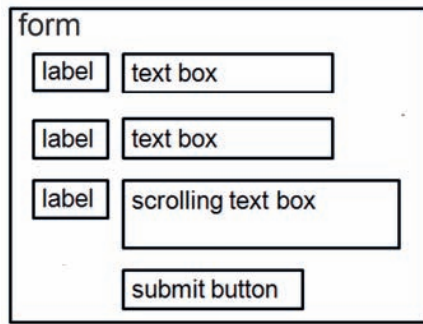


Figure 9.45 Wireframe for the form

Hands-On Practice Case Study

Task 1: Create a Folder. Create a folder called `yoga9`. Copy all of the files from your Chapter 8 `yoga8` folder into the `yoga9` folder.

Task 2: Configure the CSS. Modify the external style sheet (`yoga.css`). Open `yoga.css` in a text editor. Review Figure 9.44 and the wireframe in Figure 9.45. Notice how the text labels for the form controls are on the left side of the content area but contain right-aligned text. Notice the empty vertical space between each form control. Configure CSS as indicated below:

1. Create a label element selector to float to the left with block display. Set the text alignment to right, font-weight to bold, assign a width of 10em, and configure 1em right padding.
2. Configure the input element and textarea element selectors with block display and 2em of bottom margin.
3. Configure an id named `mySubmit` with a 12em left margin.
4. Configure the form element selector with 3em padding.

Save the `yoga.css` file.

Task 3: Create the Contact Page. Use the Home page as the starting point for the Contact page. Launch a text editor and open `index.html`. Save the file as `contact.html`. Modify your `contact.html` file to look similar to the Contact page (shown in Figure 9.44) as follows:

1. Change the page title to an appropriate phrase.
2. The Contact page will display a form in the main element. Delete all HTML and content within the main element except for the `h2` element and its text.
3. Change the text in the `h2` element to “Contact Path of Light Yoga Studio”.
4. Prepare to code the HTML for the form area. Begin with a form element that uses the post method and the action attribute to invoke server-side processing. Unless directed otherwise by your instructor, configure the action attribute to send the form data to `http://webdevbasics.net/scripts/yoga.php`.
5. Configure the form control for the Name information. Create a label element that contains the text “Name:”. Create a text box named `myName`. Use the `for` attribute to associate the label element with the form control.
6. Configure the form control for the E-mail information. Create a label element that contains the text “E-mail:”. Create a text box named `myEmail`. Use the `for` attribute to associate the label element with the form control.
7. Configure the Comments area on the form. Create a label element that contains the text “Comments:”. Create a textarea element named `myComments` with `rows` set to 2 and `cols` set to 20. Use the `for` attribute to associate the label element with the form control.

8. Configure the submit button on the form. Configure “Send Now” to display on the button. Assign the input element to the id named `mySubmit`.
9. Code an ending `</form>` tag on a blank line after the submit button.

Save your file and test your web page in a browser. It should look similar to the page shown in Figure 9.44. If you are connected to the Internet, submit the form. This will send your form information to the server-side script configured in the form tag. A confirmation page that lists the form information and their corresponding names will be displayed.

Task 4: Configure HTML5 Form Controls. Get more practice with the new HTML5 elements by modifying the form on the Contact page to use HTML5 attributes and values. Modify the `contact.html` file in a text editor.

1. Add a paragraph above the form with the following sentence: “Required information is marked with an asterisk (*).”
2. Use the `required` attribute to require the name, e-mail, and comments form controls to be entered. Add an asterisk at the beginning of each label text.
3. Configure the input element for the e-mail address with `type="email"`.

Save your file and display your web page in a browser. Submit the form with missing information or only a partial e-mail address. Depending on the browser’s level of HTML5 support, the browser may perform form validation and display an error message. Figure 9.46 shows the Contact page rendered in the Firefox browser with missing required information.

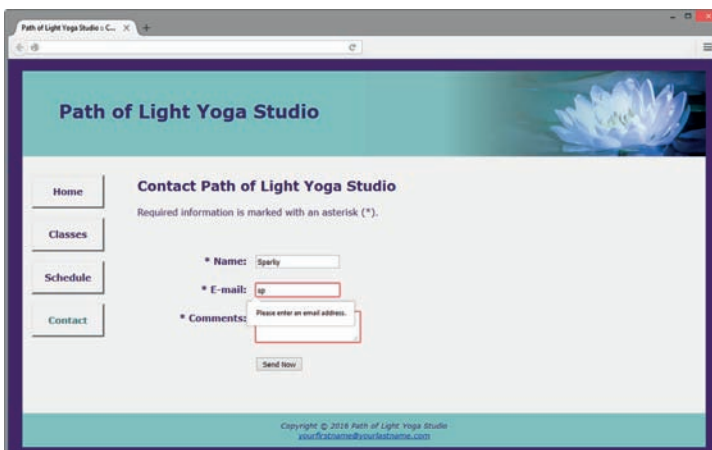


Figure 9.46 Contact page with HTML5 form controls

This task provided you with additional practice using new HTML5 attributes and values. The display and functioning of browsers will depend on the level of HTML5 support. See HTML5 Web Forms and Browser Support (<http://www.standardista.com/html5/html5-web-forms>) for an HTML5 browser support list.

Web Project

See Chapters 5 and 6 for an introduction to the Web Project case study. You will either add a form to an existing page in your website or create a new page that contains a form. Use CSS to style the form.

Hands-On Practice Case Study

1. Choose one of your project web pages to contain the form. Sketch a design of the form you plan to create.
2. Modify your project's external CSS file (project.css) to configure the form areas as needed.
3. Update your chosen web page and add the HTML code for the form.
4. The form element should use the post method and action attributes to invoke server-side processing. Unless directed otherwise by your instructor, configure the action attribute to send the form data to <http://webdevbasics.net/scripts/demo.php>.

Save and test the page. If you are connected to the Internet, submit the form. This will send your form information to the server-side script configured in the form element. A confirmation page that lists the form information and their corresponding names will be displayed.