UNIT
J
HTML5 and CSS3

# Creating and Processing Web Forms

**Files You Will Need:**

To view a list of files needed for this unit, see the Data Files Grid in the back of the book.

While Web sites are really useful for communicating information to an audience, sometimes you want to invite communications from users as well. HTML enables you to receive user input by creating **forms**, which are groups of elements that let users type text, select from lists, check boxes, and/or click buttons to provide information, and then submit that information. Phillip Blaine, the owner of Lakeland Reeds B&B, reports that customers have found his Web site useful and informative. He commonly hears that visitors would like to be able to make reservations and ask questions on the site directly. To meet this need, you design and create a feedback form.

**OBJECTIVES**

- Design a form
- Create a form
- Create text fields
- Customize text fields
- Create check boxes
- Create option buttons
- Create a drop-down menu
- Enable form submission

UNIT J

HTML5 and CSS3

# Designing a Form

Like many other parts of a Web site, it's useful to plan out a form before coding it. Understanding what information you need to collect, identifying the type of data required for each item, and ensuring that your form is logically organized and includes adequate explanations can increase usability as well as improve the accuracy and relevance of information that users provide. You met with Phillip Blaine to create a plan for the feedback form that he wants to integrate into the Lakeland Reeds B&B Web site. Figure J-1 shows a sketch of the form that you created based on the meeting. Before you finalize the form, you review some important steps in designing a form.

## DETAILS

**A few tasks are particularly important in designing a usable form:**

- **Identify the types of information you need to collect**

  Especially in larger organizations, form data can be used in many ways; for instance, the data can generate address information for sending a catalog, or the data could be used to create an account and login name for a user in an online system. Users provide most information in a form though input elements, which support different types of user interaction depending on the value of the type attribute. Table J-1 lists and describes the most commonly used values. Form elements in which users enter or select data are also known as **fields**.

  To make the data a user submits as useful as possible, it's important to ask for information in distinct pieces. For instance, if your Web form included a single field into which users entered their first and last names, you would not easily be able to sort the resulting information by last name. Providing separate input elements instead for first name and last name would enable sorting resulting records by last name. While almost every piece of information could be further broken down, it's important to clarify what you're likely to need to do with the information; thus, while you could collect street address information with separate fields for house number and street name, for most purposes, a single field for the whole address is sufficient.

- **Create a logical flow**

  A form should display related fields near each other; for example, if you were collecting name and mailing address information, you'd want to display the fields in the order that users are accustomed to specifying an address: first name, last name, street, city, etc. In addition, when you want users to complete the fields in a specific order, place the first field at the top and subsequent fields below it. Many forms end with a field where users can enter a question or additional information. Placing such a field at the end of a form invites users to first enter information in specific fields where possible, then enter in the final field anything they haven't found another place to say.

- **Integrate labels and legends**

  Fields are displayed on Web pages as boxes to check, boxes in which to enter text, or lists of options. To make the significance of each field clear to users, it's important to associate each field with a **label**, which is an element containing descriptive text that is associated with a form element.

  In most forms, groups of fields form logical units; for example, in an order form, name and address information might make up one group, details on items to order another, and payment details a third. In a Web page form, these groups are known as **fieldsets**. By default, most browsers surround the fields in a fieldset with a box, creating a visual cue that the fields share a common subject. You can further increase the usability of your form by adding a descriptive title to each fieldset. Such a title is known as a **legend** and is created using the legend element.

FIGURE J-1: Sketch of Lakeland Reeds feedback form

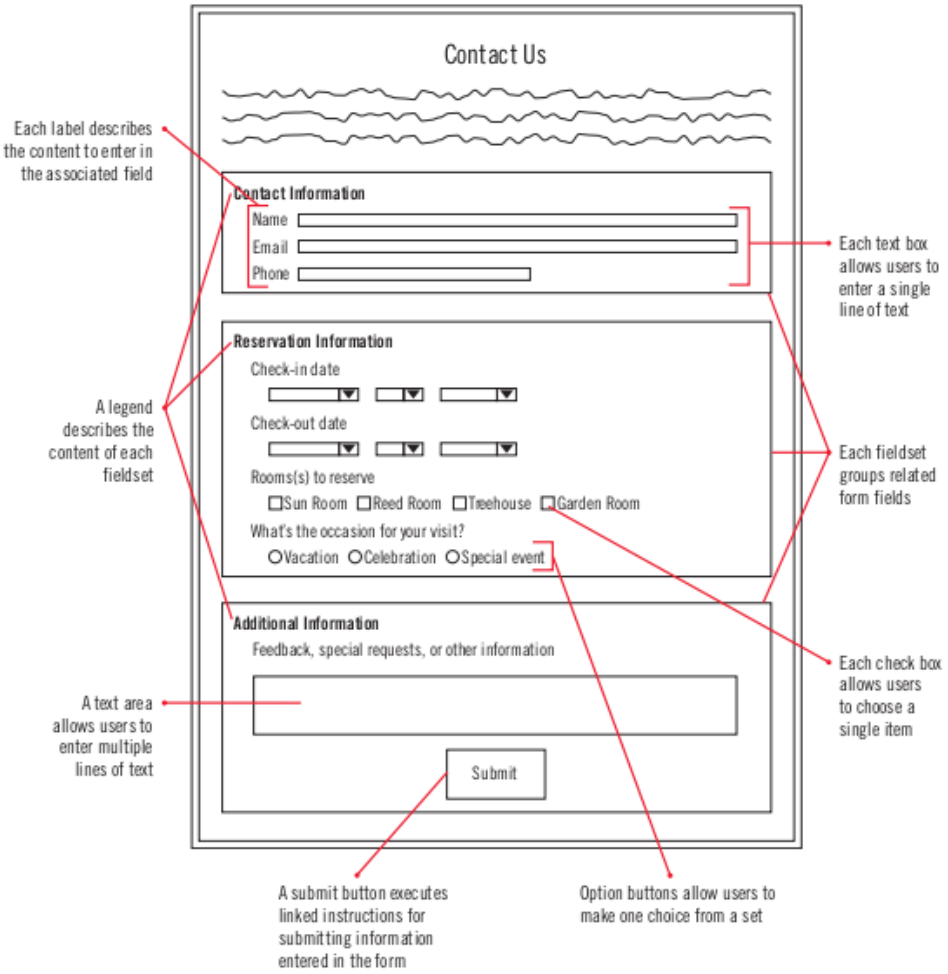



TABLE J-1: Commonly used input type values

| value | creates | example |
|---|---|---|
| checkbox | a check box, which allows users to select a single item | Treehouse |
| radio | an option button, which lets users select one option from a set of choices | Celebration |
| submit | a submit button, which users can click to send the data they have provided to the server | Submit |
| text | a text box into which users can type a single line of text | Phone |

UNIT J

HTML5 and CSS3

# Creating a Form

Like a table, a Web page form contains a series of nested elements. You mark form contents with the form element. A fieldset element contains the elements in each section of the form, including a legend element describing the contents of the fieldset. Table J-2 details some of the most commonly used form elements. As you begin to create the Lakeland Reeds B&B contact form, you enter the basic structuring elements.

## STEPS

1. In your text editor, open HTM J-1.html from the Unit J/Unit folder where you store your Data Files, insert a blank line before the closing body tag, insert a paragraph element containing your first and last name and the text HTML5 Unit J, save it as contact.html, then use CSS comments to add the same information to HTM J-2.css, saving it as lakeland.css, and to HTM J-3.css, saving it as llform.css
2. In contact.html, add a new line beneath the h2 element, indent to match the opening h2 tag, then insert opening and closing form tags on separate lines
3. Between the opening and closing form tags, add four sets of opening and closing fieldset tags on separate lines with the id values contactinfo, reserveinfo, additionalinfo, and submitbutton

**QUICK TIP**
The span elements are necessary for cross-browser styling you'll apply in a later step.

4. Within the contactinfo fieldset element, add the code <legend>Contact Information</legend>
5. Repeat Step 4 to add the legend Reservation Information to the reserveinfo fieldset and the legend Additional Information to the additionalinfo fieldset
6. Beneath the legend element in the reserveinfo fieldset, add four sets of opening and closing fieldset tags on separate lines with the id values checkin, checkout, roombox, and occasionbox

   Figure J-2 shows the completed form and fieldset tags.
7. Within the checkin fieldset, add the code <legend>Check-in date</legend>, then repeat to add the legends Check-out date to the checkout fieldset, Room(s) to reserve to the roombox fieldset, and What's the occasion for your visit? to the occasionbox fieldset

   Compare your code to Figure J-2.

**TROUBLE**
Browsers vary in the way they present fieldsets by default, so your browser may not match the figure exactly.

8. Save your work, then open contact.html in your browser

   As Figure J-3 shows, each fieldset is displayed with a border and the associated legend.
9. Return to contact.html in your text editor, copy the link element that references lakeland.css, insert a new line above the favicon link element, paste the code from the Clipboard, then change the href value in the pasted element to llform.css

   The llform.css file contains cross-browser styling that matches the rest of the Web site.
10. Save your work, then reload contact.html in your browser

    As Figure J-3 shows, the form is styled with fonts and colors that match the rest of the Web page.

**FIGURE J-2:** Structuring code for contact form

```
<article id="content">
  <h2 id="main">Contact Us</h2>
  <form>
    <fieldset id="contactinfo">
      <legend><span>Contact Information</span></legend>
    </fieldset>
    <fieldset id="reserveinfo">
      <legend><span>Reservation Information</span></legend>
      <fieldset id="checkin">
        <legend>Check-in date</legend>
      </fieldset>
      <fieldset id="checkout">
        <legend>Check-out date</legend>
      </fieldset>
      <fieldset id="roombox">
        <legend>Room(s) to reserve</legend>
      </fieldset>
      <fieldset id="occasionbox">
        <legend>What's the occasion for your visit?</legend>
      </fieldset>
    </fieldset>
    <fieldset id="additionalinfo">
      <legend><span>Additional Information</span></legend>
    </fieldset>
    <fieldset id="submitbutton">
    </fieldset>
  </form>
</article>
```

fieldset elements nested within reserveinfo fieldset

**FIGURE J-3:** Form outline in browser before and after styling

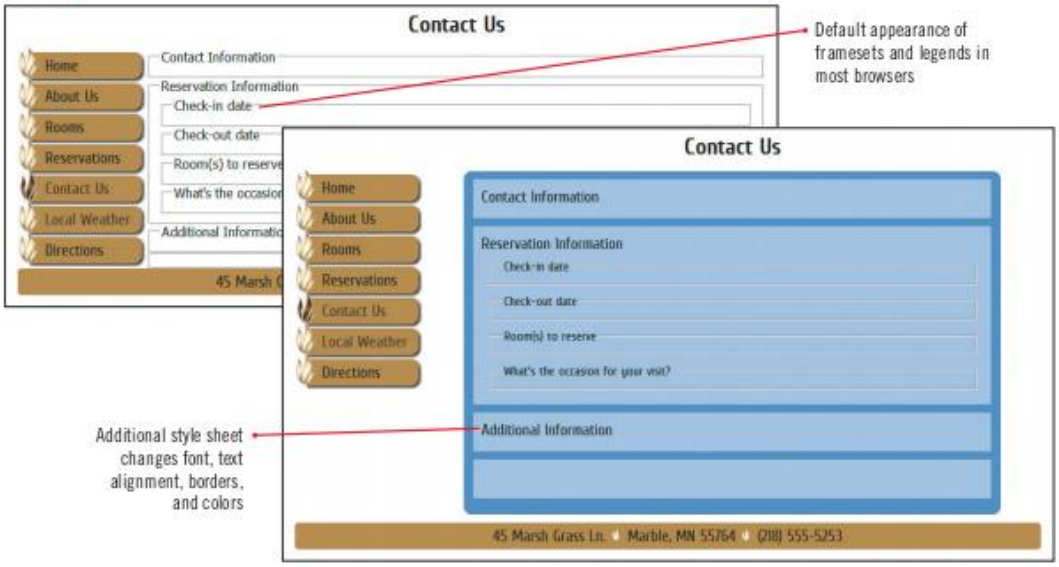



**TABLE J-2:** HTML form elements

| element | marks |
|---|---|
| fieldset | a group of related form fields and associated labels |
| form | all the elements that are part of a form |
| input | an individual item of data that users can enter |
| label | a heading describing the associated input element |
| legend | a heading describing the topic of the fieldset |
| optgroup | a group of option elements |
| option | a single entry in a drop-down list |
| select | a set of entries to display as a drop-down list |
| textarea | a multiline area where users can enter text |

UNIT J

HTML5 and CSS3

# Creating Text Fields

You can use the input element to create many different types of fields that accept user input in your forms. Setting the type attribute to "text" creates a single-line text field known as a **text box**, in which users can type a small amount of text. HTML5 introduces a number of additional input values that create text boxes with specific semantic meanings; Table J-3 details the most common values. Some user agents provide additional functionality for some of these text boxes; for instance, in an email field, newer browsers check for the common elements that all email addresses must contain and alert users if their entries are not valid email addresses. You can also use the textarea element to create a **text area**, which is a field that allows users to enter multiple lines of text. You add general text boxes for name and phone number, a special-purpose text box to collect users' email addresses, and a text area, along with legends.

## STEPS

1. Return to contact.html in your text editor
2. Within the contactinfo fieldset, beneath the legend element, enter opening and closing label tags on separate lines, then insert a new line between the tags and type Name

   Text within the label element serves as the label for the associated field.
3. Repeat Step 2 to create two more label elements below the one you just created, containing the text Email and Phone respectively, and an additional label element beneath the legend in the additionalinfo fieldset containing the text Feedback, special requests, or other information
4. Beneath the text Name in the first label element, enter the following tag:

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

   Specifying the value "text" for the type attribute creates a generic text box.
5. Beneath the text Email in the second label element, enter the following tag:

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

   The value "email" for the type attribute creates a text box and enables any special features a user agent might apply to an email field.
6. Repeat Step 4 to create a text input element with the name phone and the id phoneinput beneath the text Phone within the third label element
7. Beneath the label text in the additionalinfo fieldset, enter the code

   ```
   <textarea id="feedback" name="feedback" rows="4" cols="55"></textarea>
   ```

   The rows attribute specifies how many rows of input are visible, while the cols attribute approximates how many characters in a monospace font should fit across the box. Figure J-4 shows the completed code.
8. Save your work, then reload contact.html in your browser

   As Figure J-5 shows, the text of each label is displayed along with a text box corresponding to each input element.
9. Click in the first text box and type your first and last name, then click in the textarea box in the Additional Information section and type the text of this step to test the functionality of the box

   As you reach the end of a line in the textarea box, the text wraps, beginning a new line.

**QUICK TIP**
When a user submits a completed form, the value for each field's name attribute is paired with the user input in that field to identify the data entered in each field.

**QUICK TIP**
Although older browsers don't recognize "email" or other HTML5 input values, all browsers default to a value of "text" if no other recognized value is specified.

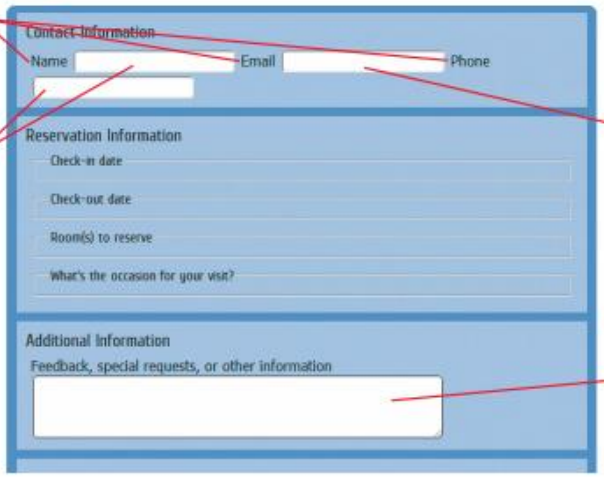
**QUICK TIP**
The textarea element uses a tag pair. Any text between the tags is displayed in the box.

**FIGURE J-4:** Text fields added to Web page

```
<form>
  <fieldset id="contactinfo">
    <legend><span>Contact Information</span></legend>
    <label>
      Name
      <input type="text" name="name" id="nameinput" />
    </label>
    <label>
      Email
      <input type="email" name="email" id="emailinput" />
    </label>
    <label>
      Phone
      <input type="text" name="phone" id="phoneinput" />
    </label>
  </fieldset>
  <fieldset id="reserveinfo">
    <legend><span>Reservation Information</span></legend>
    <fieldset id="checkin">
      <legend>Check-in date</legend>
    </fieldset>
    <fieldset id="checkout">
      <legend>Check-out date</legend>
    </fieldset>
    <fieldset id="roombox">
      <legend>Room(s) to reserve</legend>
    </fieldset>
    <fieldset id="occasionbox">
      <legend>What's the occasion for your visit?</legend>
    </fieldset>
  </fieldset>
  <fieldset id="additionalinfo">
    <legend><span>Additional Information</span></legend>
    <label>
      Feedback, special requests, or other information
      <textarea id="feedback" name="feedback" rows="4" cols="55"></textarea>
    </label>
  </fieldset>
  <fieldset id="submitbutton">
  </fieldset>
</form>
```

input and textarea elements nested within associated label elements

**FIGURE J-5:** Text fields and associated labels displayed in form




Label text entered within label elements

Text boxes created using input element with type value of "text"

Text box created using input element with type value of "email"

textarea element sized to display 4 rows of text and approximately 55 characters across

**TABLE J-3:** Input values for special data types

| value | result |
|---|---|
| password | most browsers display text entered by users as bullets or asterisks rather than showing the actual characters |
| email | newer browsers may validate to ensure that entries are valid email addresses; touchscreen devices with on-screen keyboards may display customized buttons for email input, such as an @ key or a .com key |
| url | newer browsers may validate entries to ensure that they are valid Web addresses; touchscreen devices with on-screen keyboards may display customized buttons for input, such as a .com key |
| search | browsers may style input to match styling of search boxes in other parts of the user interface |
| tel | can be used in conjunction with style sheet or script code to verify that entries are valid telephone numbers |

UNIT J

HTML5 and CSS3

# Customizing Text Fields

Labels and fields usually require styling to optimize usability. Good layout can transform a disorganized list of words and boxes into sets of labels and fields whose relationship is clear. In addition to specifying the position of label and input elements, you can control the width of text boxes and limit the number of characters users can enter in each one, as detailed in Table J-4. You can also insert attributes to input elements that add usability features in browsers that support them. You add styles to the style sheet to display the labels and text boxes in parallel columns. You also use an HTML attribute to add placeholder text to two of the fields in your form.

## STEPS

1. Return to llform.css in your text editor, create a style rule for label elements in the element with the id contactinfo, then set display to block, position to relative, and margin to 12px 0
2. Create a style rule for input elements in the element with the id contactinfo, then set position to absolute and left to 100px

   The first rule you added causes each label element to start a new line and increases the space between lines. The second rule moves all the input elements a uniform distance to the right of their corresponding labels.
3. Create a style rule for the elements with the ids nameinput and emailinput and set width to 30em, then create a style rule for the element with the id phoneinput and set width to 12em

   Compare your code to Figure J-6.
4. Save your work, then reload contact.html in your browser

   As Figure J-8 shows, the fields associated with the Name and Email labels are displayed with the same length, while the Phone text box is shorter. In addition, the labels and input boxes are displayed in separate columns.
5. Return to contact.html in your text editor, then in the input tag with the id nameinput add the name-value pair placeholder="First and last name"

   Newer browsers display the value of the placeholder attribute as light-colored text in the associated text box and remove it when a user selects the box in preparation for text entry. Placeholder text can be useful in providing examples or formats for input or for describing what users should enter in a given text box.
6. In the input tag with the id emailinput add the name-value pair placeholder="address@example.com"
7. Save your work, return to llform.css in your text editor, then add a style rule based on the input:focus and textarea:focus selectors that sets the background to #e3d5ba

   The focus pseudo-class applies to an element that a user has selected. This rule changes the background color for the field a user is working with. Figure J-7 shows the completed code.
8. Save your work, then reload contact.html in your browser

   As Figure J-8 shows, placeholder text appears in gray in the first two text boxes.
9. Click in the Name text box, type your name, then click in the Email text box

   Once you click in a text box, the placeholder text is no longer displayed, and the background color changes.

**TROUBLE**

Because some older browsers don't support the placeholder attribute, your browser may not match Figure J-8 exactly.

FIGURE J-6: Code to style text inputs and labels

```
#contactinfo label {
  display: block;
  position: relative;
  margin: 12px 0;
  }
#contactinfo input {
  position: absolute;
  left: 100px;
  }
#nameinput, #emailinput {
  width: 30em;
  }
#phoneinput {
  width: 12em;
  }
```

FIGURE J-7: Code to create placeholder text and hover color

```
<label>
  Name
  <input type="text" name="name" id="nameinput" placeholder="First and last name" />
</label>
<label>
  Email
  <input type="email" name="email" id="emailinput" placeholder="address@example.com" />
</label>
```

```
input:focus, textarea:focus {
  background: #e3d5ba;
  }
```

Values for placeholder attributes specify default text to display in text boxes

FIGURE J-8: Text boxes with positioning, size, and placeholder text applied


TABLE J-4: Sizing attributes for text boxes

| attribute | supported values | effect |
|---|---|---|
| width | width in em, pixels, or another supported unit | sets the width of the text box |
| maxlength | number greater than 0 | specifies the maximum number of characters a user can enter in the text box |

UNIT J

HTML5 and CSS3

# Creating Check Boxes

Sometimes rather than allowing users to enter text, you want to present them with a predetermined set of choices. When you want users to be able to select one or more predefined choices independent of each other, a check box usually makes the most sense. A **check box** is a box that users can click to add or remove a check mark, enabling users to select or deselect it. A check box is ideal for allowing users to indicate whether a particular statement applies to them. Because a user may wish to reserve more than one room at Lakeland Reeds, you use check boxes for inputs in the "Room(s) to reserve" fieldset.

## STEPS

1. Return to contact.html in your text editor, then beneath the legend element in the roombox fieldset, enter four pairs of label tags on separate lines
2. In the opening tag for the first label element, add the name-value pair for="sun", then repeat for the remaining three label elements using the values reed, tree, and garden
3. Within the first label element, enter the tag <input type="checkbox" id="sun" value="Sun Room" />; then on a new line type Sun Room

   Because users do not enter text in a check box, you use the value attribute to specify text to be submitted with the form if the check box is selected.

**QUICK TIP**
A check box should always precede its label text for optimal usability.

4. Repeat Step 3 for the remaining label elements, creating input elements with the following attributes and label text:

```
id="reed" value="Reed Room"        Reed Room
id="tree" value="Treehouse"        Treehouse
id="garden" value="Garden Room"    Garden Room
```

5. Add the name-value pair name="room" to each of the four input tags

   Figure J-9 shows the completed code for the check boxes and labels.
6. Save your work, then return to llform.css
7. Create a style rule for label elements within the element with the id roombox, then add a name-value pair to set the right margin to 25px

   This style adds space between each label and the check box that follows it.
8. Save your work, then reload contact.html in your browser

   Figure J-10 shows the check boxes in the Web page.
9. Click each check box to select it, then click each check box again to deselect it

   You can select as many or as few check boxes at once as you want.

**FIGURE J-9:** Code for check boxes

```
<fieldset id="roombox">
  <legend>Room(s) to reserve</legend>
  <label for="sun">
    <input type="checkbox" id="sun" value="Sun Room" name="room" />
    Sun Room
  </label>
  <label for="reed">
    <input type="checkbox" id="reed" value="Reed Room" name="room" />
    Reed Room
  </label>
  <label for="tree">
    <input type="checkbox" id="tree" value="Treehouse" name="room" />
    Treehouse
  </label for="garden">
  <label>
    <input type="checkbox" id="garden" value="Garden Room" name="room" />
    Garden Room
  </label>
</fieldset>
```

type set to "checkbox"

**FIGURE J-10:** Text boxes displayed in form

Room(s) to reserve

☐ Sun Room ☐ Reed Room ☐ Treehouse ☐ Garden Room

## Marking fields as required

Often one or more fields on a Web form are marked as required. Many Web sites implement scripts to check if required fields are completed; if not, these scripts can prevent the submission of the form and display an error message. HTML5 introduced the *required* attribute, which can replace script-based verification in some user agents. You simply add the attribute *required* (or *required="required"* in XHTML-compliant documents) to any required field. You should also include a visual cue on your Web page for each required field, along with an explanation of what the cue means. The *required* attribute can't replace script-based validation for Web sites that must support older browsers, but if your site's users primarily or exclusively access your site using newer browsers, it can be a powerful shortcut.

# Creating Option Buttons

Another type of input, the **option button**, presents users with a circular box for selecting one option from a set of choices. An option button is also known as a **radio button** and is best suited for prompting users to select only one item from a group, such as the age range that applies to users. When used appropriately, both option buttons and check boxes ensure that all user input for a particular element matches a standard list of options, preventing typographical errors and enabling you to precisely direct or sort responses that are submitted. Each input element in a set of option buttons must include the name attribute with a value identical to all other members of the set. You can also include the checked attribute for one option button in a set, causing browsers to display the button as selected by default. Phillip Blaine would like users to select only one answer for the survey question, so you use option buttons for the input elements. Because most visitors are on vacation, he'd like the vacation option to be selected by default when users open the Web page.

## STEPS

1. Return to contact.html in your text editor, then beneath the legend element in the occasionbox fieldset, enter three pairs of label tags on separate lines
2. In the opening tag for the first label element, add the name-value pair for="vacation", then repeat for the remaining two label elements using the values celebration and event
3. Within the first label element, enter the tag <input type="radio" name="occasion" id="vacation" value="Vacation" />; then on a new line type Vacation

   HTML5 allows the checked attribute with no value; for XHTML-compliant code, however, you must restate the attribute as the value to supply a full attribute-value pair.

**QUICK TIP**
Include the attribute-value pairs type="radio" and name="occasion" for all three input elements.

4. Repeat Step 3 to complete the remaining label elements having input elements with the following attributes and label text:

```
id="celebration" value="Celebration"     Celebration
id="event" value="Special Event"         Special Event
```

**QUICK TIP**
Check boxes also support the checked attribute.

5. Within the input element with the id vacation, add the name-value pair checked="checked"

   HTML5 allows the checked attribute with no value; for XHTML-compliant code, however, you must restate the attribute as the value to supply a full attribute-value pair. Figure J-11 shows the completed code for the option buttons and labels.
6. Save your work, then return to llform.css
7. In the style rule for label elements within the element with the id roombox, add a selector for label elements within the element with the id occasionbox

   This style adds space between each label and the option button that follows it.
8. Save your work, reload contact.html in your browser

   As Figure J-12 shows, the "Vacation" option button is automatically selected.
9. Click the Celebration and Special Event option buttons to select them

   As you click each button in the set, the previously selected button is deselected.

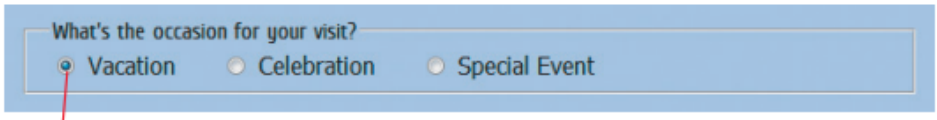
**FIGURE J-11:** Code for option buttons

```
<fieldset id="occasionbox">
  <legend>What's the occasion for your visit?</legend>
  <label for="vacation">
    <input type="radio" name="occasion" id="vacation" value="vacation" checked="checked" />
    vacation
  </label>
  <label for="celebration">
    <input type="radio" name="occasion" id="celebration" value="Celebration" />
    celebration
  </label>
  <label for="event">
    <input type="radio" name="occasion" id="event" value="special Event" />
    Special Event
  </label>
</fieldset>
```

type set to "radio"

All input elements in the set share the same value for the name attribute

**FIGURE J-12:** Option buttons displayed in form




Option selected by default using the "selected" attribute

## Implementing selection interfaces

HTML5 implemented several new input values that provide users with specific types of predefined options. Table J-5 details several of these values, along with the previously existing "file" value. All of these values create Web page features known as **selection interfaces**, which present users with allowable options visually or enable them to manipulate values without entering text. Before the HTML5 revision, a number of these interfaces were commonly created using scripts. While the HTML5 alternatives make the features easier to code, older browsers do not recognize or support these input values. Thus, if your Web page design relies on any of these input values, you'll likely need to include scripts or other field types that mimic their functions as backups in order for your pages to degrade gracefully.

**TABLE J-5:** Input values that may invoke selection interfaces

| value(s) | description | browser implementations |
|---|---|---|
| number | enables you to specify a range of valid numbers that users can input | arrows that users can click to increase or decrease the value<br>numeric virtual keyboards (touchscreen devices) |
| range | enables you to specify a range of valid numbers that users can input | a slider bar that enables users to increase or decrease the value by dragging a pointer along a line |
| date, month, week, time, datetime, datetime-local | accepts dates, times, and related values using a standard format | a calendar that users can scroll through and click on to select date-related values |
| color | supports hexadecimal color values | a color picker that visually represents colors and lets users click colors to select them |
| file | accepts the path and filename for a file to upload from a user's device | file navigation features that let users select a file from local storage |

# Creating a Drop-Down Menu

Another option for creating a list of options from which users can select is to create a **drop-down menu**, which browsers display as a small text box with a triangle next to it. Users can click the triangle to view the entire list of options; once a user clicks an option to select it, the list is hidden and the selected value is displayed in the text box. Drop-down menus are best suited to fields where you want to present a large number of options that would occupy a lot of space as option buttons. You use the select element to create a drop-down menu, with option elements nested within it to specify the list items. Phillip Blaine wants to give users who want to make reservations the option to specify arrival and departure dates. You create a drop-down menu for the date, month, and year for both dates.

## STEPS

1. In your text editor open the file htmj-4.txt, select all the contents, copy them to the Clipboard, then close the file
2. Return to contact.html in your text editor, add a new line beneath the legend element in the checkin fieldset, then paste the contents of the Clipboard
   You'll use these lists of dates, months, and years to create your drop-down menus.
3. Add opening and closing select tags around the list of months, specifying the value inmonth for the id and name attributes, repeat for the list of dates using the value indate, then repeat for the list of years using the value inyear
4. In the list of months, click before the word January, type <option value="01">, click after the word January, type </option>, then repeat to add option elements to the remaining months, assigning the value 02 to February, 03 to March, and so on
5. In the list of dates, click before the number 1, type <option value="01">, click after the number 1, type </option>, then repeat to add option elements to the remaining dates, assigning the value 02 to 2, 03 to 3, and so on
6. In the list of years, click before 2013, type <option value="2013">, click after 2013, type </option>, then repeat to add option elements to the remaining years, assigning a value attribute corresponding to each year
   Figure J-13 shows the completed code for the check-in dates.
7. Copy the three select elements to the Clipboard, insert a blank line below the legend element in the checkout fieldset, paste the contents of the Clipboard, then change the id and name values for the select elements you pasted to outmonth, outdate, and outyear, respectively
   You'll offer the same date options to users for selecting a check-out date.
8. Save your work, then reload contact.html in your browser
   As Figure J-14 shows, the first option element within each select element is displayed as the default choice.
9. Use the drop-down menus to select a check-in date of August 3, 2015 and a check-out date of August 6, 2015

**QUICK TIP**
You can add the "selected" attribute to an option element in order to make it selected by default.

FIGURE J-13: Code for check-in drop-down menus

Each select element creates a drop-down menu

Each option element defines a single menu option

```
<select id="inmonth" name="inmonth">
  <option value="01">January</option>
  <option value="02">February</option>
  <option value="03">March</option>
  <option value="04">April</option>
  <option value="05">May</option>
  <option value="06">June</option>
  <option value="07">July</option>
  <option value="08">August</option>
  <option value="09">September</option>
  <option value="10">October</option>
  <option value="11">November</option>
  <option value="12">December</option>
</select>
<select id="indate" name="indate">
  <option value="01">1</option>
  <option value="02">2</option>
  <option value="03">3</option>
  <option value="04">4</option>
  <option value="05">5</option>
  <option value="06">6</option>
  <option value="07">7</option>
  <option value="08">8</option>
  <option value="09">9</option>
  <option value="10">10</option>
  <option value="11">11</option>
  <option value="12">12</option>
  <option value="13">13</option>
  <option value="14">14</option>
  <option value="15">15</option>
  <option value="16">16</option>
  <option value="17">17</option>
  <option value="18">18</option>
  <option value="19">19</option>
  <option value="20">20</option>
  <option value="21">21</option>
  <option value="22">22</option>
  <option value="23">23</option>
  <option value="24">24</option>
  <option value="25">25</option>
  <option value="26">26</option>
  <option value="27">27</option>
  <option value="28">28</option>
  <option value="29">29</option>
  <option value="30">30</option>
  <option value="31">31</option>
</select>
<select id="inyear" name="inyear">
  <option value="2013">2013</option>
  <option value="2014">2014</option>
  <option value="2015">2015</option>
</select>
```

FIGURE J-14: Drop-down menus for check-in and check-out dates

Drop-down menus fit long lists of choices into a small area of the Web page


## Creating option groups

Especially in a long set of options, it's sometimes helpful to group the options in a drop-down menu. For instance, if you were presenting users with a list of countries, grouping the countries by continent could help a user more quickly identify the part of the list where their country is located. To create an option group, you enclose a group of option elements within an *optgroup* element. You then use the *label* attribute to add heading text to the group. In most browsers, options are indented below the label for their group, creating a hierarchical list that can simplify navigation.

# Enabling Form Submission

A form needs to include a **submit button**, which is a button that users can click to submit the data they've entered. You can use the "submit" input type to create a standard submit button. Table J-6 details two other input values that also create buttons. In the opening form tag, you add the "action" attribute to specify the name and location of a script on your Web server to accept the form data, and the "method" attribute to indicate the way the data should be submitted. User agents group the name attribute of each field with the value entered or selected by a user; thus, every field with information to be submitted must have a value for the name attribute. Figure J-15 shows an example of what submitted user data from the contact form might look like. You'll work later with other members of the Web site team to set up and test a server script to process the data. For now, you create a submit button.

## STEPS

1. Return to contact.html in your text editor
2. Within the submitbutton fieldset, enter the following code:
```
<input type="submit" id="submit" value="Submit" />
```
The value attribute specifies the text displayed on the button.
3. Save your work, then reload contact.html in your browser
The submit button is displayed at the bottom of the form. By default, form buttons are usually gray with black text and a subtle box shadow.
4. Return to contact.html in your text editor, then just above the closing </head> tag, enter the following code:
```
<!--[if IE]>
  <style type="text/css">
    legend span {top: 0;}
    legend {color: black;}
    #submit {position: relative; left: -80px;}
  </style>
<![endif]-->
```
Internet Explorer lays out form elements differently than other browsers. This conditional comment creates an embedded style sheet for this Web page, but only if the browser opening the files identifies itself as IE.
5. Save your work, return to llform.css in your text editor, add the two style rules shown in Figure J-16, save your work, then reload contact.html in your browser
As Figure J-17 shows, the submit button now matches the style of the rest of the form.
6. Create a print version of the main style sheet with the name llprint.css and a print version of the form style sheet with the name llformpr.css, then in contact.html, add a link element referencing llformpr.css and specifying a media value of print
7. Validate the code for contact.html and your style sheets, then make changes as necessary to fix any errors
8. If you have space on a Web server, publish your Web site, then open contact.html in your browser from the published location

FIGURE J-15: Example of data from a submitted form

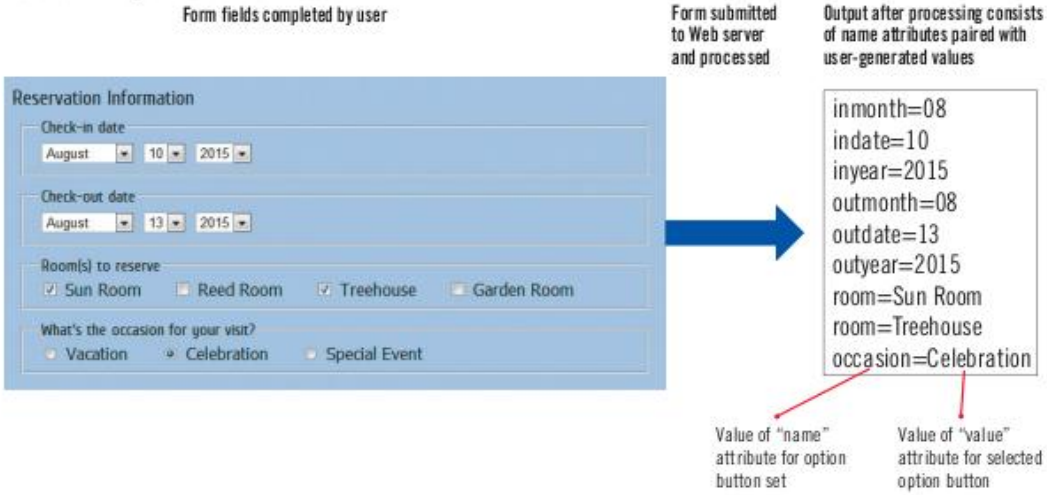



FIGURE J-16: Code to style submit button

```
#submit {
  background: #e3d5ba;
  font-size: 1.25em;
  }
#submitbutton {
  border: none;
  background: #6a91ca;
  padding: 0.5em 0 0 0;
  text-align: center;
  }
```

FIGURE J-17: Completed contact form


TABLE J-6: Input values for buttons

| value | description | attributes |
|---|---|---|
| **submit** | creates a default submit button that submits user input based on form or button attributes | **value** specifies button text |
| **image** | creates a submit button using an image | **src** specifies the image file name and location<br>**alt** provides alternative text for users of non-visual user agents |
| **reset** | clears all user input and resets fields to defaults; not used by some designers because users can confuse it with a submit button and lose all input | **value** specifies button text |
| **button** | creates a generic button that can be programmed using a script | value specifies button text |

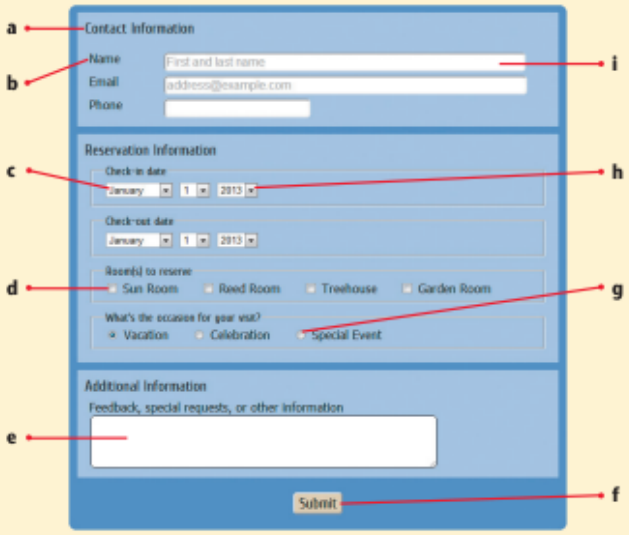
# Practice

For current SAM information, including versions and content details, visit SAM Central (http://www.cengage.com/samcentral). If you have a SAM user profile, you may have access to hands-on instruction, practice, and assessment of the skills covered in this unit. Since various versions of SAM are supported throughout the life of this text, check with your instructor for the correct instructions and URL/Web site for accessing assignments.

## Concepts Review

**Refer to Figure J-18.**

FIGURE J-18




1. **Which item is created with a select element?**
2. **Which item is created with an option element?**
3. **Which element is created using an input type of "checkbox"?**
4. **Which element is created using an input type of "radio"?**
5. **Which element is created using an input type of "text"?**
6. **Which item is created with a textarea element?**
7. **Which item is created with a legend element?**
8. **Which item is created with a label element?**
9. **Which element is created using an input type of "submit"?**

**Match each term with the statement that best describes it.**

| | |
|---|---|
| **10. field** | **a.** a single-line text field |
| **11. label** | **b.** a button that users can click to submit the data they've entered |
| **12. fieldset** | **c.** a descriptive title for a fieldset |
| **13. legend** | **d.** a form element in which users enter or select data |
| **14. text box** | **e.** a box that users can click to add or remove a check mark |
| **15. check box** | **f.** a group of fields that form a logical unit |
| **16. option button** | **g.** descriptive text associated with a form element |
| **17. text area** | **h.** a box for selecting one option from a set of choices |
| **18. submit button** | **i.** a field that allows users to enter multiple lines of text |

**Select the best answer from the list of choices.**

**19. Which is an example of asking users for information in distinct pieces?**

**a.** using a single text box for first and last name
**b.** using a text area for general questions
**c.** separating date entry into month, date, and year fields
**d.** creating custom text for a submit button

**20. Which element is used to explain the purpose of other form elements?**

**a.** input
**b.** legend
**c.** option
**d.** textarea

**21. Newer browsers display the value of the _____ attribute as light-colored text in the associated text box and remove it when a user selects the box in preparation for text entry.**

**a.** placeholder
**b.** type
**c.** legend
**d.** label

**22. Which pseudo-class applies to an element that a user has selected?**

**a.** hover
**b.** focus
**c.** active
**d.** followed

**23. Each input element in a set of option buttons must include the _____ attribute with a value identical to all other members of the set.**

**a.** selected
**b.** id
**c.** value
**d.** name

**24. Which element do you use to create a drop-down menu?**

**a.** fieldset
**b.** textarea
**c.** input
**d.** select

**25. When submitting a form, user agents group the _____ attribute of each field with the value entered or selected by a user.**

**a.** name
**b.** id
**c.** placeholder
**d.** value

## Skills Review

**1. Create a form.**

**a.** In your text editor, open HTM J-5.html from the Unit J/Review folder where you store your Data Files, insert a blank line before the closing body tag, insert a paragraph element containing your first and last name and the text **HTML5 Unit J**, and save it as **order.html**, then use CSS comments to add the same information to HTM J-6.css, saving it as **bigj.css**, and to HTM J-7.css, saving it as **bigjform.css**.

**b.** In order.html, add a new line beneath the h2 element, indent to match the opening h2 tag, then insert opening and closing form tags on separate lines.

**c.** Between the opening and closing form tags, add four sets of opening and closing fieldset tags on separate lines with the id values **deliveryinfo**, **orderinfo**, **additionalinfo**, and **submitbutton**.

**d.** Within the deliveryinfo fieldset element, add the text **Delivery Information** within a span element, then enclose the span element within a legend element. Repeat, including the span element, to add the legend **Order** to the orderinfo fieldset and the legend **Special Instructions** to the additionalinfo fieldset.

**e.** Beneath the legend element in the orderinfo fieldset, add three sets of opening and closing fieldset tags on separate lines with the id values **crustbox**, **sizebox**, and **toppingbox**.

**f.** Within the crustbox fieldset, add a legend element containing the text **Crust**, then repeat to add the legends **Size** to the sizebox fieldset and **Topping(s)** to the toppingbox fieldset. Save your work, then open order.html in your browser.

**g.** Return to order.html in your text editor, copy the link element that references bigj.css, insert a new line above the favicon link element, paste the code from the Clipboard, then change the href value in the pasted element to **bigjform.css**. Save your work, then reload order.html in your browser.

## Skills Review (continued)

2. **Create text fields.**
   a. Return to order.html in your text editor.
   b. Within the deliveryinfo fieldset, beneath the legend element, add a label element containing the text **Name**. Repeat to create four more label elements below the one you just created, containing the text **Street Address**, **City**, **Email**, and **Phone**, respectively, then add an additional label element beneath the legend in the additionalinfo fieldset containing the text **Special requests, delivery details**.
   c. Beneath the text Name in the first label element, enter an input tag for a text field with the name **name** and id **nameinput**, then add a name-value pair to the opening label tag associating it with the input element you created. Repeat to create a text field below the text Street Address with the name **address** and id **addrinput**, a text field below the text City with the name **city** and id **cityinput**, an email field below the text Email with the name **email** and id **emailinput**, and a text field below the text Phone with the name **phone** and id **phoneinput**.
   d. Beneath the label text in the additionalinfo fieldset, insert an element to create a text area with the name **instructions** and id **instructions** that displays **3** rows and **60** columns of input. Add a name-value pair to the opening label tag, associating it with the text area you created.
   e. Save your work, then reload order.html in your browser. Enter text in each of the fields you just created.
3. **Customize text fields.**
   a. Return to bigjform.css in your text editor, create a style rule for label elements in the element with the id deliveryinfo, then set the display to **block**, the position to **relative**, and the margin to **12px 0**.
   b. Create a style rule for input elements in the element with the id deliveryinfo, then set position to **absolute** and left to **120px**.
   c. Create a style rule for the elements with the ids nameinput, emailinput, and addrinput, and set the width to **30em**, then create a style rule for the elements with the ids cityinput and phoneinput, and set the width to **12em**.
   d. Save your work, then reload order.html in your browser.
   e. Return to order.html in your text editor, then in the input tag with the id nameinput, add **First and last name** as placeholder text. Repeat to add **Building number and street** to the input tag with the id addrinput and **address@example.com** to the input tag with the id emailinput. Save your work.
   f. Return to bigjform.css in your text editor, then add a style rule that sets the backgrounds of the input and textarea elements to **#f99** when users select them.
   g. Save your work, then reload order.html in your browser. Verify that the placeholder text is displayed in the fields where you added it, then click in each text field and verify that the background color changes.
4. **Create check boxes.**
   a. Return to order.html in your text editor, then beneath the legend element in the toppingbox fieldset, enter five pairs of label tags on separate lines.
   b. In the opening tag for the first label element, add an attribute-value pair to associate it with the element having the id **pepperoni**. Within the label element, enter code to create an input element for a check box with the id **pepperoni** and value **Pepperoni**. Add a new line beneath the input element and enter the text **Pepperoni**.
   c. Repeat Step b to complete the remaining label elements with input elements with the following attributes and label text:

| id | value | label text |
|---|---|---|
| sausage | Sausage | Sausage |
| greenpep | Green Peppers | Green Peppers |
| onion | Onions | Onions |
| xcheese | Extra Cheese | Extra Cheese |

   d. Add the name attribute with the value **toppings** to each of the five input tags.
   e. Save your work, then return to bigjform.css.
   f. Create a style rule for label elements within the element having the id **toppingbox**, then add a name-value pair to set the right margin to **25px**.

## Skills Review (continued)

g. Save your work, then reload order.html in your browser. Click each check box to select it, then click each check box again to deselect it.

5. **Create option buttons.**
   a. Return to order.html in your text editor, then beneath the legend element in the crustbox fieldset, enter two pairs of label tags on separate lines.
   b. In the opening tag for the first label element, add an attribute-value pair to associate it with the element having the id **thin**. Within the label element, enter code to create an input element for an option button with the id **thin** and value **Thin**. Add a new line beneath the input element and enter the text **Thin Crust**.
   c. Repeat Step b to associate the second label element with the element having the id **thick** and to create an input element for an option button with the id **thick**, value **Deep Dish**, and label text **Deep Dish**. Add an attribute to make this option selected by default. Save your work, then return to bigjform.css.
   d. In the style rule for label elements within the element with the id toppingbox, add a selector for **label** elements within the element with the id **crustbox**.
   e. Save your work, then reload order.html in your browser. Click the Thin Crust and Deep Dish option buttons to select them.
6. **Create a drop-down menu.**
   a. Return to order.html in your text editor, add a new line beneath the legend element in the sizebox fieldset, then add opening and closing tags for a select element with the id and name **size**.
   b. Add an option element with a value attribute set to **small** and the content **Small**. Repeat to create option elements with a value of **medium** and content **Medium**, a value of **large** and content **Large**, and a value of **XL** and content **Extra Large**.
   c. Save your work, then reload order.html in your browser. Use the drop-down menu to select each of the size options.
7. **Enable form submission.**
   a. Return to order.html in your text editor.
   b. Within the submitbutton fieldset, enter code for an input element that creates a submit button with an id of **submit** and value of **Add to Cart**.
   c. Save your work, then return to bigjform.css in your text editor. Add a style rule for the element with the id submit that sets the background to **red** and the font size to **1.25em**. Add another style rule for the element with the id submitbutton that removes the border, sets the background to **black**, sets padding to **0.5em 0 0 0**, and center-aligns text.
   d. Save your work, then reload order.html in your browser. Compare your screen to Figure J-19.
   e. Create a print version of the main style sheet with the name **bjprint.css** and a print version of the form style sheet with the name **bjformpr.css**, then in order.html, add a link element referencing **bjformpr.css** and specifying a media value of **print**.
   f. Validate the code for order.html and your style sheets, then make changes as necessary to fix any errors.
   g. If you have space on a Web server, publish your Web site, then open order.html in your browser from the published location.

FIGURE J-19

## Independent Challenge 1

The Spotted Wren Garden Center has begun offering free landscaping consultations. Sarah Nguyen, the owner, would like to add a form to the Web site to enable users to request an appointment. You create and style the form.

**a.** Open HTM J-8.html from the Unit J/IC1 folder in your text editor. Insert a blank line before the closing body tag, insert a paragraph element containing your first and last name and the text **HTML5 Unit J, Independent Challenge 1**, and save it as **quote.html**, then use a CSS comment to add the same information to **HTM J-9.css**, saving the file as **spotwren.css**.

**b.** In quote.html, insert tags for a form element before the closing article tag. Within the form element, add tags for five fieldset elements with the id values **contactinfo**, **timinginfo**, **jobtype**, **additionalinfo**, and **submitbutton**. Save your work, then in spotwren.css, add a style rule for the form element to set the background color to **#6c6** and specify a padding of **20px** at the top of the element. Add a style rule for fieldset elements that sets the bottom margin to **12px**, the element width to **90%**, and the margins to **0 auto**.

**c.** In the fieldset with the id contactinfo, add the legend **Contact Info** followed by five label elements. Within the first label element, add the label text **Name**, then add code to create a text box with the id **nameinput** and the name **name**. Add an attribute to the opening label tag to associate it with the input element you created. Repeat for the remaining label elements, using the following values:

| id | name | label text |
|---|---|---|
| addrinput | address | Street Address |
| zipinput | zip | Zip Code |
| emailinput | email | Email |
| phoneinput | phone | Phone |

**d.** Edit the emailinput element to specify that the expected data is an email address. Add the placeholder text **First and last name** to the nameinput element and **address@example.com** to the emailinput element.

**e.** Save your work, then in spotwren.css, add a style rule that sets the font weight for legend elements to **bold**. Add another style rule that applies to label elements within the element with the id contactinfo, setting the display to **block**, the position to **relative**, and the margin to **12px 0**. Create a style rule that applies to input elements within the element having the id contactinfo, setting the position to **absolute** and **130px** to the left. Create another rule that sets the widths of the elements with the ids nameinput, emailinput, and addrinput to **30em**, and an additional rule that sets the widths of the elements with the ids zipinput and phoneinput to **12em**.

**f.** In the fieldset with the id timinginfo, add the legend **Best day(s) to schedule a visit** followed by seven label elements. Within the first label element, add code to create a check box with the id **monday**, the name **days**, and the value **Monday**, then add the label text **Monday**. Add an attribute to the opening label tag to associate it with the input element you created. Repeat for the remaining label elements to create check boxes for the other six days of the week in chronological order.

**g.** In the fieldset with the id jobtype, add the legend **Project Area** followed by four label elements. Within the first label element, add code to create an option button with the id **front**, the name **area**, and the value **House front**, then add the label text **Front of House**. Add an attribute to the opening label tag to associate it with the input element you created. Repeat for the remaining label elements, using the same name and the following additional values:

| id | value | label text |
|---|---|---|
| border | Border | Border of Property |
| multiple | Multiple | Multiple Areas (please specify in Notes box below) |
| other | Other | Other (please specify in Notes box below) |

## Independent Challenge 1 (continued)

**h.** Save your work, then in spotwren.css, add a style rule for label elements in the element with the id jobtype that sets the display to **block**. Save your changes to spotwren.css.

**i.** In the fieldset with the id additionalinfo, add the legend **Additional Information** followed by a label element. Within the label element, add the label text **Notes**, then add code to create a text area with the id and name **notes**, displaying **4** rows and **60** columns of input. Add an attribute to the opening label tag to associate it with the text area element you created.

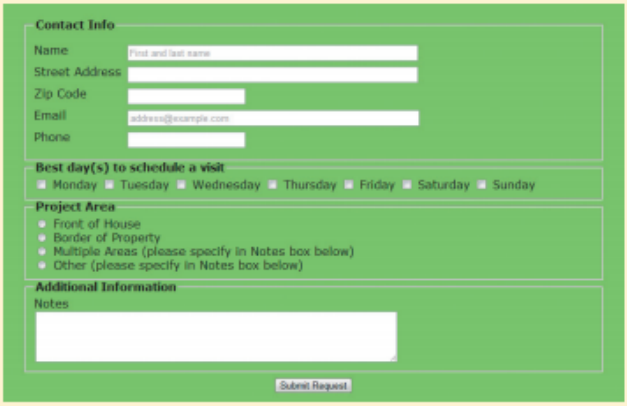
**j.** In the fieldset with the id submitbutton, add code to create a submit button with the id **submit** and the value **Submit Request**. Save your work. In spotwren.css, create a style rule that applies to the element with the id submitbutton and that removes the border, center-aligns the text, and removes the bottom margin. Save your changes to spotwren.css.

**k.** Add a style rule that applies to currently selected input and textarea elements, setting the background to **#ff0**. Save your work, preview quote.html in your browser, then compare your screen with Figure J-20.

**l.** Create a print version of the stylesheet, saving it with the name **swprint.css**, then validate your HTML and CSS documents. If you have space on a Web server, publish your Web site, then open resource.html in your browser from the published location.

**m.** Close your browser, then close your text editor.

FIGURE J-20




## Independent Challenge 2

The coordinators of the Murfreesboro Recreational Soccer League would like to allow prospective new members to sign up on the Web site. You create a form to allow users to enter basic information.

**a.** Open HTM J-10.html from the Unit J/IC2 folder in your text editor. Insert a blank line before the closing body tag, insert a paragraph element containing your first and last name and the text **HTML5 Unit J, Independent Challenge 2**, save the file as **signup.html**, then use a CSS comment to add the same information to HTM J-11.css and save it as **mrsl.css**.

**b.** In signup.html, below the h2 element, add a form element containing five fieldsets. Add the following ids to the fieldsets: **contactinfo**, **agebox**, **membershipbox**, **additionalinfo**, **submitbutton**. Add legends to the first four fieldsets containing the text **Contact Information**, **Age Range**, **Special Memberships**, and **Additional Information**. Save your work, then in mrsl.css, add a style rule to set the width of the form to **60%** and the margin to **0 auto**. Add another style rule to set the fieldset borders to **2px solid black**, and a third to present the text of legends in **bold**. Save your changes to mrsl.css.

**c.** In the first fieldset, add four label elements. Add the label text **First Name**, **Last Name**, **Email**, and **Phone**. Add a text input field below the label text in the first label element, using the id **fnameinput** and the name **fname**. Add an attribute to the label tag to associate it with this element. Repeat to add a text input field to the second label element with the id **lnameinput** and the name **lname**, an email field to the third label element with the id **emailinput** and the name **email**, and a text input field to the fourth label element with the id **phoneinput** and the name **phone**. Specify **First name** as placeholder text for the first text field, **Last name** for the second, and **address@example.com** for the third. Save your work.

## Independent Challenge 2 (continued)

**d.** In mrsl.css, add a style rule setting the background for input and textarea elements to **#f99** when users select them. Add a style rule for labels in the contactinfo element, setting the display to **block**, the position to **relative**, and the margin to **12px 0**. Create another style rule for input elements within the contactinfo element, setting position to **absolute** and **130px** to the left. Create style rules to set the widths of the emailinput element to **30em**, fnameinput and lnameinput elements to **20em**, and the phoneinput element to **12em**. Save your changes to mrsl.css.

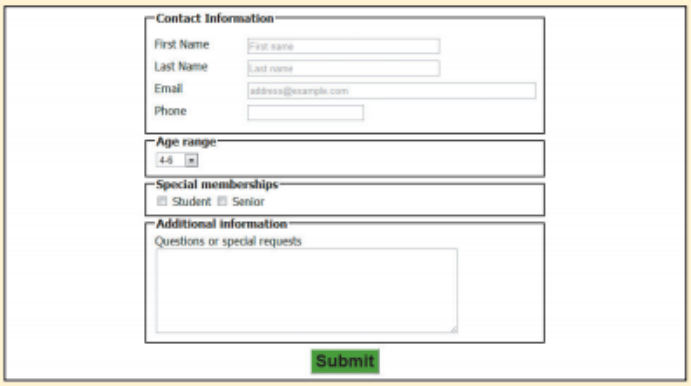
**e.** In the fieldset with the id agebox, add a select element with the id and name **agerange**. Create eight options for the drop-down list, displaying the text **4-6**, **7-9**, **10-12**, **13-15**, **16-17**, **18-35**, **36-54**, and **55+**. Set the value for the first option to **D1**, the second to **D2**, and so on through the final element, which will have the value **D8**.

**f.** In the fieldset with the id membershipbox, add two label elements. Within the first label element, add code to create a check box with the id **student**, the name **memberships**, and the value **Student**, then add the label text **Student**. Add an attribute to the opening label tag to associate it with the input element you created. Repeat for the second label element to create a check box with the id **senior**, the name **memberships**, and the value **Senior**, then add the label text **Senior**.

**g.** In the fieldset with the id additionalinfo, add a label element. Within the label element, add the label text **Questions or special requests**, then add code to create a text area with the id and name **feedback**, displaying **7** rows and **50** columns of input. Add an attribute to the opening label tag to associate it with the text area element you created. Save your work, then in mrsl.css, add a style rule for the element with the id feedback and setting the display to **block**. Save your changes to mrsl.css.

**h.** In the fieldset with the id submitbutton, add code to create a submit button with the id **submit** and the value **Submit**. Save your work. In mrsl.css, create a style rule that applies to the element with the id submitbutton, removing the border and center-aligning text. Create a style rule for the element with the id submit, setting the background to **#090**, the font size to **1.5em**, and the font weight to **bold**. Save your changes to mrsl.css. Open signup.html in your Web browser and compare your screen to Figure J-21.

FIGURE J-21




**Advanced Challenge Exercise**

- In the fnameinput and lnameinput elements, add the attribute-value pair **required=required**.
- Below the h2 element, insert a new line containing a paragraph element with the text **Fields marked with * are required.** Add the id **formexpl** to the opening paragraph tag.
- After the label text First name, type a space followed by *****. Repeat for the label text Last name.
- In the paragraph element and the two labels, enclose the * with a span element having the class **req**. Save your work.
- In mrsl.css, add a style rule for the req class, setting the color to **red**. Add another style rule for the id formexpl, setting the width to **60%** and the margin to **0 auto**. Save your work.
- Preview the Web page in your browser, then click the Submit button without entering text in either of the name fields. (*Note:* Not all browsers check for required fields.)

## Independent Challenge 2 (continued)

i. Create a print version of the style sheet, saving it with the name **mrsprint.css**, then validate your HTML and CSS documents.
j. If you have space on a Web server, publish your Web site, then open signup.html in your browser from the published location.
k. Close your browser, then close your text editor.

## Independent Challenge 3

Diego Merckx, the manager of the Hotel Natoma, wants to allow guests to initiate reservations using the Web site. You create a Web page form based on the information he wants to collect from potential visitors.

a. Open HTM J-12.html from the Unit J/IC3 folder in your text editor. Insert a blank line before the closing body tag, then insert a paragraph element containing your first and last name and the text **HTML5 Unit J, Independent Challenge 3**. Save the file as **reserve.html**, then use CSS comments to add the same information to HTM J-13.css, saving it as **natoma.css**, and HTM J-14.css, saving it as **hnform.css**.
b. In reserve.html, add a form element below the h2 element, then add four fieldsets. Add the legend text **Contact Information**, **Reservation Information**, and **Additional Information** to the first three fieldsets. Add an appropriate id to each fieldset (the fourth fieldset will contain the Submit button). Within the fieldset having the legend Reservation Information, nest five additional fieldsets, add the legends **Party size**, **Bed preference**, **Check-in date**, **Check-out date**, and **Parking**, then add an appropriate id to each fieldset.
c. Save your work, then preview the Web page in a browser. In reserve.html, add a link element that references the file **hnform.css**, specifying a media type of **screen**. Save your work, then refresh the page in your browser.
d. In reserve.html, add three label elements to the first fieldset, containing the label text **Name**, **Email**, and **Phone**. Add code for a text box within each label element, using the relevant input types and appropriate name and id values. Add placeholder text if appropriate. Associate each label element with the enclosed text box. Save your work, then in hnform.css, add styles to set the display for the labels to **block**, the position to **relative**, and adding a **12px** margin to the left and right sides. Add styles to set the position for the text boxes in the first fieldset to **absolute** and specifying a left value that moves all the boxes to the right of the label text; add styles to set an appropriate width for each text box, then add styles to set the background color of selected input and textarea elements to **#e3d5ba**. Save your work, then reload the Web page in your browser.
e. In the fieldset with the legend Party size, add two label elements containing the label text **Number in your party** and **Number of rooms required**. Add code for a text box within each label element using appropriate name and id values. Associate each label element with the enclosed text box. Save your work, then in hnform.css, add styles to add a right margin of **20px** to both of the text boxes you created and specify a width of **3em** for both. Save your work, then reload the Web page in your browser.
f. In the fieldset with the legend Bed preference, add three label elements containing the label text **King/Queen**, **Twin**, and **Other/Mix (please specify in Additional Information section below)**. Add code for a check box before the label text within each label element, using appropriate name and id values. Associate each label element with the enclosed check box. Save your work, then in hnform.css, add styles to add a right margin of **25px** to each check box element. Save your work, then reload the Web page in your browser.
g. In the fieldset with the legend Check-in date, paste the contents of htmj-15.txt. Create three drop-down lists, using the pasted text as list items to create lists of months, dates, and years. Add appropriate ids and names to the lists. Copy the code for the three drop-down lists and paste it into the fieldset with the legend Check-out date, then change the ids and names for the pasted items to unique values. Save your work, then in hnform.css, add styles to make the Check-in date fieldset float to the **left** and the Check-out date fieldset float to the **right**, and to set the width of each element to **45%** of the parent. Save your work, then reload the Web page in your browser.

## Independent Challenge 3 (continued)

**h.** In the fieldset with the legend Parking, add a label element containing the label text **I need parking for**. Add code for a text box within the label element using appropriate name and id values. Associate the label element with the enclosed text box. After the code for the text box, add the text **vehicles**. Save your work, then in hnform.css, add styles to set the width of the text box to **3em**. Save your work, then reload the Web page in your browser.

**i.** In the fieldset containing the legend text Additional Information, add a label element. Within the label element, add the label text **Feedback, special requests, or other information**, then add code to create a text area with an appropriate id and name, displaying **4** rows and **55** columns of input. Associate the label element with the text area element. Save your work, then in hnform.css, add styles to set the display for both the label and text area elements to **block**. Save your changes, then reload the Web page in your browser.

**j.** In the final fieldset, add code to create a submit button with an appropriate id and the value **Submit**. Save your work. In hnform.css, create a style rule that applies to the fieldset, removing the border and center-aligning text. Add styles setting the background of the submit button to **#e3d5ba**, font size to **1.25em**, and a **10px** border radius to the upper-left and lower-right corners. Save your work, then reload the Web page in your browser and compare your screen to Figure J-22.

FIGURE J-22

**Advanced Challenge Exercise**

- Change the input type for the text boxes in the Party size and Parking fieldsets from text to **number**. In the input element with the label Number in your party, add the attribute-value pairs **min="1"** and **max="50"**. In the input element with the label Number of rooms required, add the attribute-value pairs **min="1"** and **max="38"**. In the input element beneath the legend Parking, add the attribute-value pairs **min="0"** and **max="22"**. Save your work, reload the Web page in your browser, and explore any formatting or tools the browser may have added to the number fields.
- In the opening tag for the form element, add a unique id value, then in hnform.css, change the selector for the form style rule to reflect the new id. Add a new form element below the nav element containing the main nav bar, assigning it a unique id. Add a label element to the form containing the text **Google**, then below it, add a text box using the input type **search**. Add appropriate name and id values and associate the label element with the search box. Below the label element, add code for a submit button with a value of **Search**.
- In the opening tag for the new form element, add the code **action="http://www.google.com/search" method="get"**. Save your work, then in hnform.css, add a style rule for the new form element, then add styles to give the form an **absolute** position **10px** from the right and **80px** from the top of the closest ancestor element with positioning. Add another style rule to set the width of the text box to **170px**. Save your work, then reload the Web page in your browser and note if your browser formats the search box at the top of the page differently than other text boxes. In the search box, type **HTML5**, then click the Search button.

**k.** Create print versions of the style sheets, saving them with the names **hnprint.css** and **hnfprint.css**, add a link element to reserve.html referencing hnfprint.css for printed output, save your work, then validate your HTML and CSS documents.

**l.** If you have space on a Web server, publish your Web site, then open reserve.html in your browser from the published location.

**m.** Close your browser, then close your text editor.

## Real Life Independent Challenge

This assignment builds on the personal Web site you have worked on in previous units. You'll plan and add a form to your site.

**a.** Copy your Web site files to the Unit J/RLIC folder on the drive and folder where you store your Data Files.

**b.** On paper or in a new document in your text editor, list specific information you'd like users to be able to submit from your Web site. Break each piece of information down to an appropriate size to work with once it's submitted.

**c.** Group the information you've listed into categories. For each piece of information, identify the most appropriate form element to use in collecting it. Sketch the form you want to create, including fieldsets containing related fields.

**d.** If necessary, create a new Web page for the form and update the nav bar on each of the site's pages to include the new page.

**e.** Add a form element, followed by the fieldset elements necessary to structure your form. Next add fields and labels. Finally, add style rules to your style sheet document to optimize the form layout and increase usability. Implement placeholder text and/or background color for fields that users click. Save your work, then preview your form in a browser.

**f.** Create a print version of your style sheet, save your work, then validate all of your HTML and CSS documents and make any edits necessary to address any issues.

**g.** If you have space on a Web server, publish your Web site, then open the page containing your form in your browser from the published location.

**h.** Close your browser, then close your text editor.

## Visual Workshop

In your text editor, open the file HTML J-16.html from the Unit J/VW directory on the drive and folder where you store your Data Files, add a paragraph before the closing body tag that contains your first and last name and the text **HTML5 Unit J, Visual Workshop**, save the file as **signup.html**, then use CSS comments to add the same information to **HTM J-17.css**, saving it as **revision.css**. Use your text editor to add a form to make your Web page match Figure J-23. Save your work, save a copy of your style sheet with the name **revprint.css**, then make any necessary changes for print formatting. When you are finished, validate your HTML and CSS code. If you have space on a Web server, publish your files, then open the Web page in your browser from the published location. Close your browser and text editor.

FIGURE J-23