HTML5, CSS3, and JavaScript

6th Edition

Tutorial 7

Designing a Web Form

Carey

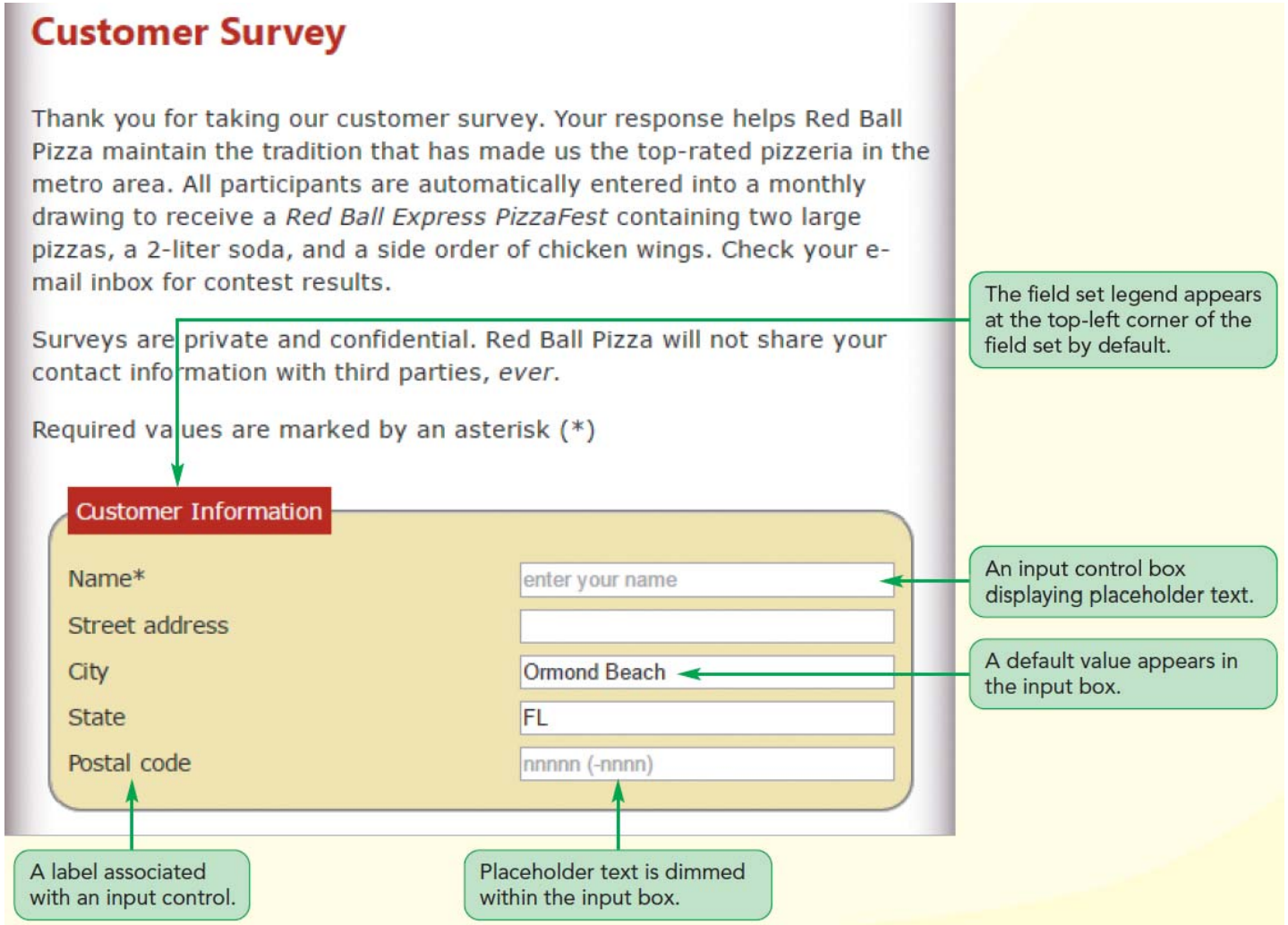
Objectives

- Explore web forms
- Work with form servers
- Create forms and field sets
- Create labels and input boxes
- Explore form layout
- Work with date and time fields
- Create a selection list

Objectives (continued)

- Create option buttons
- Create check boxes and text area boxes
- Create spinners and range sliders
- Use data lists
- Create form buttons
- Validate a form
- Apply validation styles

Structure of a Web Form



Introducing Web Forms

- **Web form**
  - Allows users to enter data that can be saved and processed
  - Common way to accept user input
  - Allows the creation of interactive websites for user feedback

Parts of a Web Form

- **Controls,** also known as **widgets,** are the objects that allow a user to interact with a form
- Each data entry control is associated with a **data field**
- **Data field**: Stores the data values supplied by a user

Parts of a Web Form (continued 1)

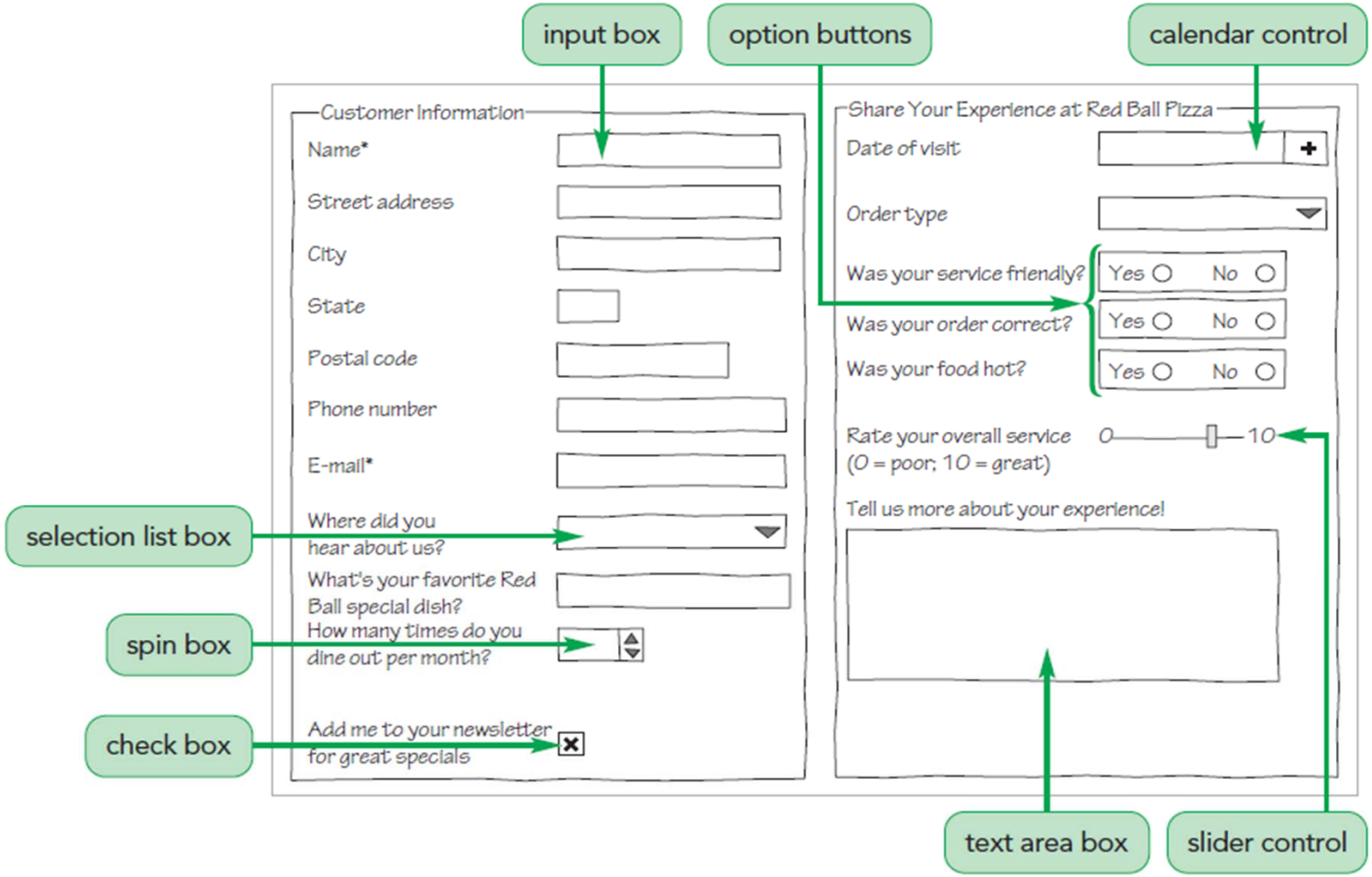
- Types of **controls**
  - **Input boxes** to insert text and numeric values
  - **Option/radio buttons** to select data values from a predefined set of options
  - **Selection lists** to select data values from an extensive list of options
  - **Check boxes** to select data values limited to two possibilities, such as “yes” or “no”
  - **Text area boxes** to enter text strings that may include several lines of content

Parts of a Web Form (continued 2)

- Types of **widgets**
  - **Spin boxes** to enter integer values confined to a specified range
  - **Slider controls** to enter numeric values confined to a specified range
  - **Calendar controls** to select date and time values
  - **Color pickers** to choose color values

Parts of a Web Form (continued 3)

Figure 7-1 Proposed survey form

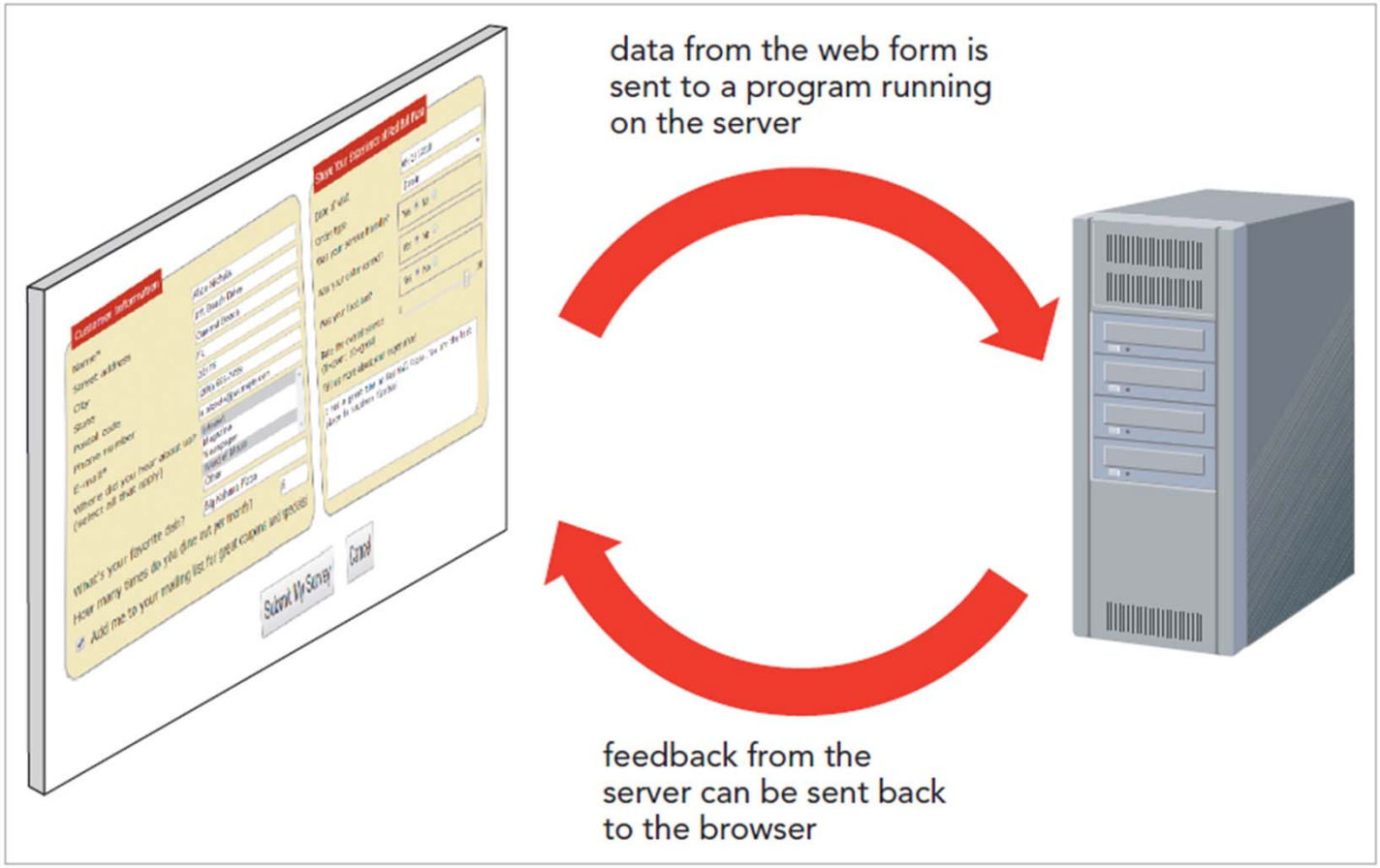


Forms and Server-Based Programs

- Field values entered by a user are processed by a program running on the user's computer or on a web server in a secure location
- Example: A web form is used to collect data from a customer for an order and the server program processes the data and handles the billing and delivery

Forms and Server-Based Programs (continued)

Figure 7-2 Interaction between the web form and the server



Starting a Web Form

- Web forms are marked using the `form` element

```
<form id="text" attributes>
    content
</form>
```

  - `id` identifies the form
  - *`attributes`* specify how the form should be processed by the browser
  - *`content`* is the form's content

Starting a Web Form (continued)

- A `form` element can be placed anywhere within the body of a page
- Forms also can contain page elements such as tables, paragraphs, inline images, and headings

Figure 7-3 Inserting a web form

form id identifies the form

```
<p>Required values are marked by an asterisk (*)</p>

<form id="survey">
</form>
```

form element marks the web form contents

Interacting with the Web Server

- The `action`, `method`, and `enctype` attributes have to be included in a form to specify where and how to send the form data

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

Interacting with the Web Server (continued 1)

- `action` attribute provides the location of the web server program that processes the form
- `method` attribute specifies how the browser should send form data to the server
- `enctype` attribute specifies how the form data should be encoded as it is sent to the server

Interacting with the Web Server (continued 2)

- Two possible values for `method` attribute
  - **Get method**: Tells the browser to append the form data to the end of the URL specified in the `action` attribute
  - The **get method** is the default method
  - **Post method**: Sends the form data in its own separate data stream
  - The **post method** is considered to be a more secure form of data transfer

Interacting with the Web Server (continued 3)

- The `enctype` attribute has three possible values

Figure 7-4 Values of the enctype attribute

Value	Description
application/x-www-form-urlencoded	The default format in which the data is encoded as a long text string with spaces replaced by the + character and special characters (including tabs and line breaks) replaced with their hexadecimal code values
multipart/form-data	The format used when uploading files in which no encoding of the data values occurs
text/plain	The format in which data is transferred as plain text with spaces replaced with the + character but no other encoding of the data values occurs

Interacting with the Web Server (continued 4)

Figure 7-5 Associating the web form with an action and a method

```
<p>Required values are marked by an asterisk (*)</p>

<form id="survey" action="http://www.example.com/redball/survey" method="post">
</form>
```

location of server processing the form

method by which the form data is sent to the server

Interacting with the Web Server (continued 5)

- A script element is an HTML element used to access and run JavaScript programs that will run within the user's browser

Figure 7-6 Using a script to manage the form submission

```
<title>Red Ball Survey</title>
<link href="rb_reset.css" rel="stylesheet" />
<link href="rb_styles.css" rel="stylesheet" />
<script src="rb_formsubmit.js"></script>
</head>
```

the script element runs JavaScript programs within the browser

external JavaScript file

Creating a Field Set

- **Field set**: Groups fields that share a common purpose
- Field sets are created using the `fieldset` element

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

  - *id* identifies the field set
  - *content* is the form content within the field set

Marking a Field Set

Figure 7-7 Inserting field sets

```
<form id="survey" action="http://www.example.com/redball/survey" method="post">
   <fieldset id="custInfo">
   </fieldset>

   <fieldset id="expInfo">
   </fieldset>
</form>
```

id associated with each fieldset element

Adding a Field Set Legend

- Legend describes the content of a field set using the `legend` element

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

  where *`text`* is the text of the legend
- The `legend` element contains only text and no nested elements
- By default, legends are placed in the top-left corner of the field set box and can be moved to a different location using the CSS positioning styles

Adding a Field Set Legend (continued)

Figure 7-8 Adding legends to the field sets

legend associated with each field set

```
<form id="survey" action="http://www.example.com/redball/survey" method="post">
   <fieldset id="custInfo">
      <legend>Customer Information</legend>
   </fieldset>

   <fieldset id="expInfo">
      <legend>Share Your Experience at Red Ball Pizza</legend>
   </fieldset>
</form>
```

Creating Input Boxes

- Syntax for the `input` element

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

  - The `name` attribute provides the name of the data field associated with the control
  - The `id` attribute identifies the control in which the user enters the field value
  - The `type` attribute indicates the data type of the field

Creating Input Boxes (continued 1)

Figure 7-10 Controls and the input type attribute

Type Value	Control Displayed by the Browser
button	A button that can be clicked to perform an action
checkbox	A check box for yes/no or true/false responses
color	A widget from which users can select a color
date	A widget from which users can select a calendar date
datetime-local	A widget from which users can select a calendar date and time
email	An input box used for e-mail addresses
file	A widget from which users can select a local file
hidden	A control that is hidden from the user
image	An image that can be clicked to perform an action
month	A widget from which users can select a calendar month and year
number	A spin box from which users can select a numeric value
password	An input box in which the entry value is hidden by * symbols
radio	A radio or option button that can be clicked by the user
range	A slider from which users can select a numeric value within a defined range
reset	A button that can be clicked to reset the web form
search	A widget that can be used to search for a defined term
submit	A button that can be clicked to submit the form for processing
tel	An input box used for telephone numbers
text (the default)	An input box used for text entries
time	A widget from which users can select a time value
url	An input box used for entering URLs
week	A widget from which users can select a week value

Creating Input Boxes (continued 2)

Figure 7-11 Adding input elements to the form

name of the field associated with the input box

id of the input box control for entering the customer name

input boxes for general text entries

input box for telephone numbers

input box for e-mail addresses

```
<fieldset id="custInfo">
   <legend>Customer Information</legend>
   Name*
   <input name="custName" id="name" type="text" />

   Street address
   <input name="custStreet" id="street" type="text" />

   City
   <input name="custCity" id="city" type="text" />

   State
   <input name="custState" id="state" type="text" />

   Postal code
   <input name="custZip" id="zip" type="text" />

   Phone number
   <input name="custPhone" id="phone" type="tel" />

   E-mail*
   <input name="custEmail" id="mail" type="email" />
</fieldset>
```

Input Types and Virtual Keyboards

- **Virtual keyboards** are software representations of a physical device
- Web forms can be made responsive by displaying different virtual keyboards for each input type
- Example: An input box for telephone number is more convenient to read with digits displayed prominently on the keyboard

Adding Field Labels

- To associate a text string with a control, the text string has to be enclosed within the `label` element

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

  - *`id`* is the id of the control that is associated with the label
  - *`label text`* is the text of the label

Adding Field Labels (continued)

Figure 7-14 Adding form labels

for attribute associates the label with the name input box

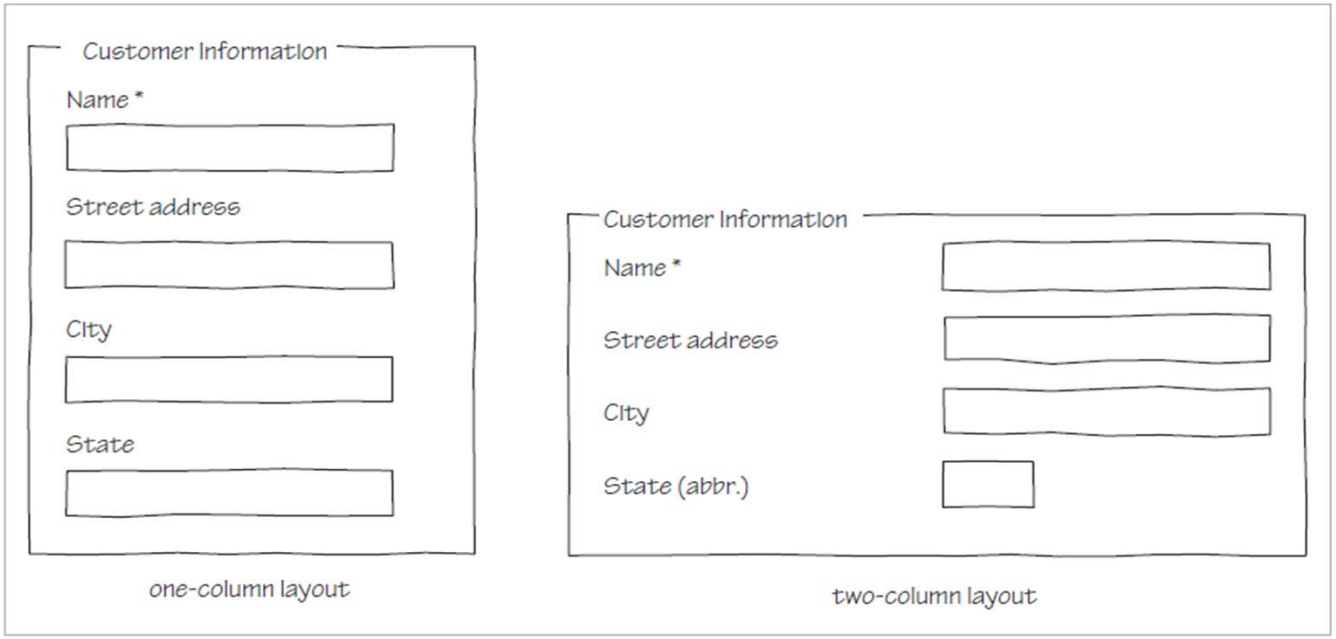
label element

```
<legend>Customer Information</legend>
<label for="name">Name*</label>
<input name="custName" id="name" type="text" />

<label for="street">Street address</label>
<input name="custStreet" id="street" type="text" />

<label for="city">City</label>
<input name="custCity" id="city" type="text" />

<label for="state">State</label>
<input name="custState" id="state" type="text" />

<label for="zip">Postal code</label>
<input name="custZip" id="zip" type="text" />
```

Designing a Form Layout

- There are two general layouts
  - Labels are placed directly above the input controls
  - Labels and controls are placed side-by-side

Figure 7-15 Form layouts



Defining Default Values and Placeholders

- The `value` attribute is used to specify a default field value

Figure 7-22 Defining the default field value

```
<div class="formRow">
   <label for="city">City</label>
   <input name="custCity" id="city" type="text" value="Ormond Beach" />
</div>

<div class="formRow">
   <label for="state">State</label>
   <input name="custState" id="state" type="text" value="FL" />
</div>
```

sets the default value for the custCity field

sets the default value for the custState field

Defining Default Values and Placeholders (continued 1)

- **Placeholders:** Text strings that appear within a form control, providing a hint about the kind of data that should be entered into a field
- They are defined using the `placeholder` attribute

  `placeholder="`*`text`*`"`

  where *`text`* is the text of the placeholder

Defining Default Values and Placeholders (continued 2)

Figure 7-23 Defining placeholder text

```
<div class="formRow">
   <label for="name">Name*</label>
   <input name="custName" id="name" type="text" placeholder="first and last name" />
</div>
```

placeholder text for the name input box

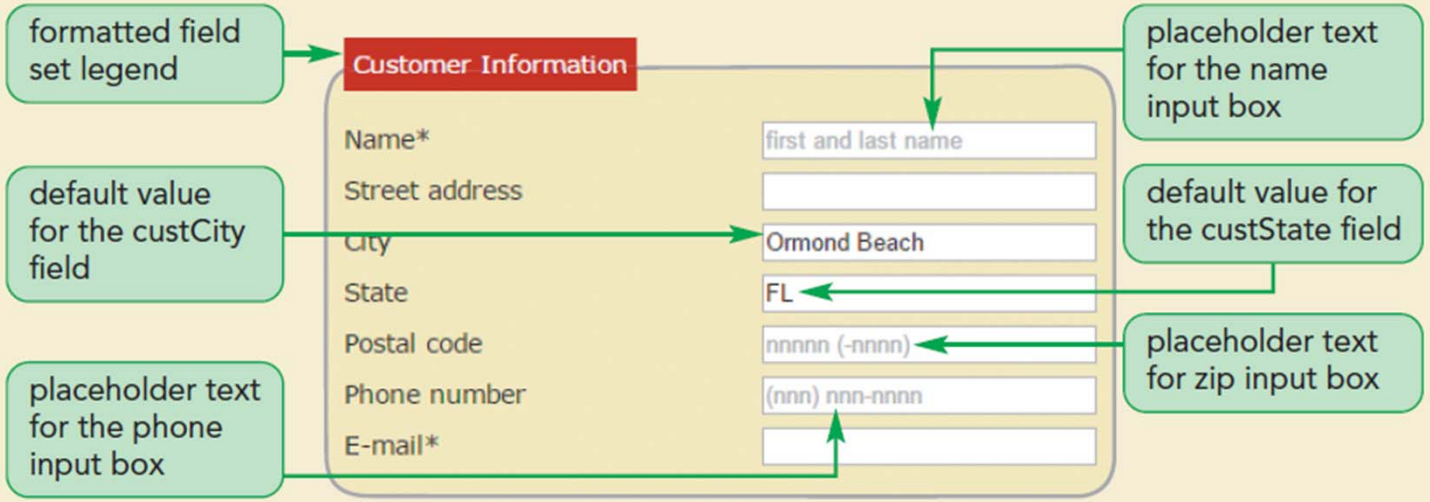
```
<div class="formRow">
   <label for="zip">Postal code</label>
   <input name="custZip" id="zip" type="text" placeholder="nnnnn (-nnnn)" />
</div>

<div class="formRow">
   <label for="phone">Phone number</label>
   <input name="custPhone" id="phone" type="tel" placeholder="(nnn) nnn-nnnn" />
</div>
```

placeholder text for the zip input box

placeholder text for the phone input box

Defining Default Values and Placeholders (continued 3)

Figure 7-24 Viewing default values and placeholder text



Entering Date and Time Values

- Date and time fields ensure that users enter data in the correct format
- Indicated using `type` attributes: `date, time, datetime-local, month, and week`

Figure 7-26 Creating a date field

```
<fieldset id="expInfo">
   <legend>Share Your Experience at Red Ball Pizza</legend>

   <div class="formRow">
      <label for="visit">Date of visit</label>
      <input name="visitDate" id="visit" type="date" />
   </div>

</fieldset>
```

sets the data type of the visitDate field to "date"

Creating a Selection List

- A selection list is a list box that presents users with a group of possible values for the data field
- The list is created using the `select` and `option` elements

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

Creating a Selection List (continued 1)

- *`name`* is the name of the data field
- *`value1`*, *`value2`*,… are the possible field values
- *`text1`*, *`text2`*,… are the text of the entries in the selection list that users see on the web form

Creating a Selection List (continued 2)

Figure 7-27 Creating a selection list for the orderType field

```
<div class="formRow">
   <label for="visit">Date of visit</label>
   <input name="visitDate" id="visit" type="date" />
</div>

<div class="formRow">
   <label for="order">Order type</label>
   <select name="orderType" id="order">
      <option value="order1">Carry out</option>
      <option value="order2">Delivery</option>
      <option value="order3" selected>Dine in</option>
      <option value="order4">Take 'n bake</option>
   </select>
</div>
```

field name associated with the selection list

id of the selection list control

order3 (Dine in) is the default selected value of the orderType field

possible values of the orderType field

text strings displayed in the selection list for each option

Working with Select Attributes

- By default, a selection list appears as a drop-down list box
- To display a selection list as a scroll box, use the `size` attribute to the `select` element

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

  where *value* is the number of options that the selection list displays at one time

Working with Select Attributes (continued 1)

- By default, selection lists allow only one selection from the list of options
- To allow more than one item to be selected, add `multiple` attribute

```
<select multiple> ... </select>
```

Working with Select Attributes (continued 2)

- Two ways for users to select multiple items from a selection list
  - For non-contiguous selection, press and hold the Ctrl key while making the selections
  - For contiguous selection, select the first item, press and hold the Shift key, and then select the last item in the range

Grouping Selection Options

- Organize selection list options by placing them in option groups using the `optgroup` element

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

where *label1* is the label for the different groups of options

Grouping Selection Options (continued)

Figure 7-31 Grouping options in a selection list

```
<label for="appetizers">Starter Menu</label>
<select name="meal">
   <optgroup label="Appetizers">
      <option value="sms">Spicy Mozzarella Sticks</option>
      <option value="pr">Pepperoni Rolls</option>
      <option value="tr">Toasted Ravioli</option>
   </optgroup>
   <optgroup label="Salads">
      <option value="sms">Pasta Salad</option>
      <option value="tbs">Tuscan Bread Salad</option>
      <option value="pr">Caesar Salad</option>
   </optgroup>
</select>
```

option group labels

Creating Option Buttons

- Option buttons are also called radio buttons
- Unlike selection lists, the options appear as separate controls in the web form
- They are created with a group of `input` elements with a `type` attribute value of "radio"

```
<input name="name" value="value1"
type="radio" />
```

Creating Option Buttons (continued 1)

```
<input name="name" value="value2"
type="radio" />
…
```

where *name* is the name of the data field and *value1*, *value2*, *value3*,… are the field values associated with each option

- Set an option button to be selected as the default by adding the `checked` attribute to the input element
- `<input name="name" type="radio" checked />`

Creating Option Buttons (continued 2)

Figure 7-32 Creating option groups for the sFriend, oCorrect, and foodHot fields

```
<div class="formRow">
   <label>Was your service friendly?</label>
   <fieldset class="optGroup">
      <label for="fYes">Yes</label>
      <input name="sFriend" id="fYes" value="yes" type="radio" />
      <label for="fNo">No</label>
      <input name="sFriend" id="fNo" value="no" type="radio" />
   </fieldset>
</div>

<div class="formRow">
   <label>Was your order correct?</label>
   <fieldset class="optGroup">
      <label for="cYes">Yes</label>
      <input name="oCorrect" id="cYes" value="yes" type="radio" />
      <label for="cNo">No</label>
      <input name="oCorrect" id="cNo" value="no" type="radio" />
   </fieldset>
</div>

<div class="formRow">
   <label>Was your food hot?</label>
   <fieldset class="optGroup">
      <label for="hYes">Yes</label>
      <input name="foodHot" id="hYes" value="yes" type="radio" />
      <label for="hNo">No</label>
      <input name="foodHot" id="hNo" value="no" type="radio" />
   </fieldset>
</div>

   </fieldset>
</form>
```

label for the option group

options for the sFriend field

field value

radio button controls

options for the oCorrect field

options for the foodHot field

Creating Check Boxes

- Check boxes are designed for fields that record the presence or absence of an object or event
- They are created using the `input` element with the `type` attribute set to "`checkbox`"

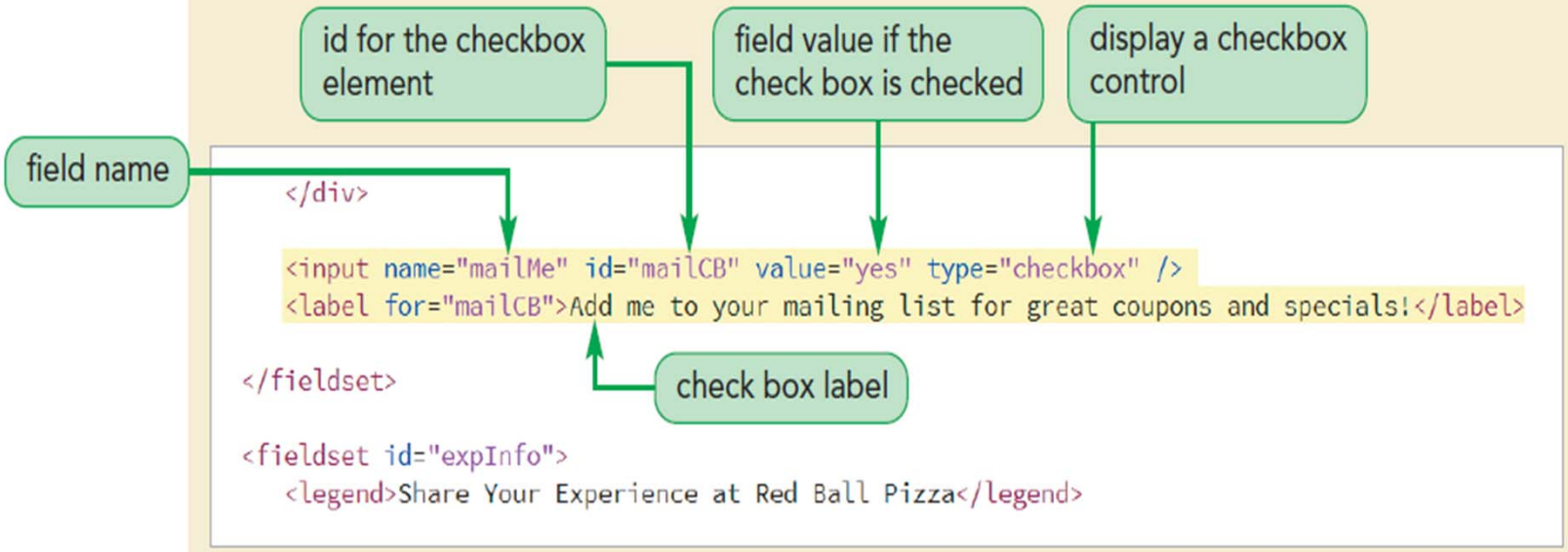
```
<input name="name" value="value"
type="checkbox" />
```

  - *`value`* attribute contains the value of the field when the check box is checked
  - *`type`* attribute indicates that the input box is a check box

Creating Check Boxes (continued)

- By default, a check box is not checked

Figure 7-34 Creating a checkbox control




```
    </div>

    <input name="mailMe" id="mailCB" value="yes" type="checkbox" />
    <label for="mailCB">Add me to your mailing list for great coupons and specials!</label>

  </fieldset>

  <fieldset id="expInfo">
    <legend>Share Your Experience at Red Ball Pizza</legend>
```

Creating a Text Area Box

- Text area is created using the `textarea` element

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

where *text* is the default value of the data field

Creating a Text Area Box (continued 1)

- HTML supports the `rows` and `cols` attributes to set the text area size

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

  - `rows` attribute specifies the number of lines in the text area box
  - `cols` attribute specifies the number of characters per line

Creating a Text Area Box (continued 2)

Figure 7-36 Creating a text area box

```
      </div>

      <label for="commBox">Tell us more about your experience!</label>
      <textarea name="custExp" id="commBox"></textarea>

   </fieldset>
</form>
```

displays a text area box

field name

id of the text area box

Figure 7-38 Text area box in the web form

Entering Numeric Data

- **Creating a Spinner Control**
  - **Spinner control:** Displays an up or down arrow to increase or decrease the field value by a set amount
  - To create a spinner control, apply the `input` element using the `number` data type

Entering Numeric Data (continued 1)

Figure 7-39 Creating a spinner control for the dineOut field

```
<div class="formRow">
   <label for="dineSpin">How many times do you dine out per month?</label>
   <input name="dineOut" id="dineSpin" type="number"
          value="1" step="1" min="0" max="20" />
</div>

<input name="mailMe" id="mailCB" value="yes" type="checkbox" />
<label for="mailCB">Add me to your mailing list for great coupons and specials!</label>
```

displays a spinner control

the field name

the default field value

the minimum and maximum values of the field

the amount by which the field value increases in the spinner

Entering Numeric Data (continued 2)

- **Creating a Range Slider**
  - **Slider control:** Limits a numeric field to a range of possible values
  - To create a slider control, apply the range data type in the `input` element

Entering Numeric Data (continued 3)

Figure 7-42 Creating a range slider control for the serviceRate field

```
<div class="formRow">
   <label for="rangeBox">Rate the overall service<br />
         (0=poor; 10=great)</label>
   0
   <input name="serviceRate" id="rangeBox" type="range"
         value="5" step="1" min="0" max="10" />
   10
</div>

<label for="commBox">Tell us more about your experience!</label>
<textarea name="custExp" id="commBox"></textarea>
```

field name

displays a range slider control

text strings that will appear before and after the range slider

default value of the serviceRate field

step size between values on the range slider

minimum and maximum values for the serviceRate field

Suggesting Options with Data Lists

- **Data list:** A list of possible data values that a form field can have
- Data lists are defined using the `datalist` element

```
<datalist id="id">
    <option value="value" />
    <option value="value" />
…
</datalist>
```

Suggesting Options with Data Lists (continued)

Figure 7-44 Applying a data list to the favDish field

```
<div class="formRow">
   <label for="dish">What's your favorite dish?</label>
   <input name="favDish" id="dish" type="text" list="dishType" />
   <datalist id="dishType">
      <option value="Anitpasto Pizza" />
      <option value="Big Kahuna Pizza" />
      <option value="BBQ Chicken Pizza" />
      <option value="Mediterranean Herb Pizza" />
      <option value="Pasta Rolls" />
      <option value="Pasto Artichoke Pizza" />
   </datalist>
</div>

<div class="formRow">
   <label for="dineSpin">How many times do you dine out per month?</label>
   <input name="dineOut" id="dineSpin" type="number"
          value="1" step="1" min="0" max="20" />
</div>
```

data list containing suggested values

links the favDish field to the dishType data list

Working with Form Buttons

- **Form buttons:** A type of form control that performs an action
- Actions performed
  - Run a command from a program linked to the web form
  - Submit the form to a program running on the web server
  - Reset the form fields to their default values

Creating a Command Button

- **Command button:** Runs a program that affects the content of a page or the actions of a browser
- Created using the `input` element with the `type` attribute set to `button`

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

  - *`text`* is the text that appears on the button
  - *`script`* is the name of the program code that is run when the button is clicked by the user

Creating Submit and Reset Buttons

- **Submit button:** Submits a form to the server for processing when clicked
- Submit button is created using input elements with the `type` attribute set to "submit" and "reset" respectively

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

where *`text`* is the text string that appears on the button

Creating Submit and Reset Buttons (continued 1)

- **Reset button:** Resets a form, changing all fields to their default values and deleting any field values that a user has entered
- Reset button is created using input elements with the `type` attribute set to "reset"

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

where *`text`* is the text string that appears on the button

Creating Submit and Reset Buttons (continued 2)

Figure 7-46 Creating submit and reset buttons

```
   </fieldset>

   <div id="buttons">
      <input type="submit" value="Submit My Survey" />
      <input type="reset" value="Cancel" />
   </div>
</form>
```

creates a submit button

creates a reset button

the text on the button control

Designing a Custom Button

- The appearance of a command, submit, and reset button is determined by the browser
- For more control over a button's appearance use the `button` element

```
<button type="text">
        content
</button>
```

  - `type` attribute specifies the button type
  - *`content`* are HTML elements placed within the button

Validating a Web Form

- **Validation:** Process of ensuring that a user has supplied valid data
- Types of validation
  - **Server-side validation** – validation occurs on the web server
  - **Client-side validation** – validation occurs in the user's browser

Identifying Required Values

- The first validation test is to verify if data is supplied for all the required data fields
- Add the `required` attribute to the control to identify the required data fields
- If a required field is left blank, the browser will not submit the form returning an error message to indicate the unavailability of data

Validating Based on Data Type

- A form fails the validation test if the data values entered into a field do not match the field type
- Example:
  - A data field with the `number` type will be rejected if nonnumeric data is entered
  - Fields with `email` and `url` types will be rejected if a user provides an invalid e-mail address or text that does not match the format of a URL

Testing for a Valid Pattern

- To test whether a field value follows a valid pattern of characters, test the character string against a regular expression
- **Regular expression** or **regex** is a concise description of a character pattern
- To validate a text value against a regular expression, add the `pattern` attribute to the `input` element

Testing for a Valid Pattern (continued)

- Example: The value of the custZip field against the regular expression pattern `^\d{5}$`

```
<input name="custZip"
pattern="^\d{5}$" />
```

where regular expression `^\d{5}$` represents any string of five numeric characters

Defining the Length of the Field Value

- The syntax to define the `maxlength` attribute is `<input maxlength="value" />`

  where *value* is the maximum number of characters in the field value
- Example:

```
<input name="custZip" maxlength="5" />
```

- The `maxlength` attribute does not distinguish between characters and digits

Applying Inline Validation

- **Inline validation:** The technique of immediate data validation and reporting of errors
- The `focus` pseudo-class is used to change the display style of fields that currently contain invalid data
- **Focus:** The state in which an element has been clicked by the user, making it the active control on the form

Applying Inline Validation (continued 1)

Figure 7-54 Pseudo-classes for form controls and fields

Pseudo-Class	Matches
`checked`	A check box or option button that is selected or checked
`default`	A default control, such as the default option in a selection list
`disabled`	A control that is disabled
`enabled`	A control that is enabled
`focus`	A control that has the focus (is actively selected) in the form
`indeterminate`	A check box or option button whose toggle states (checked or unchecked) cannot be determined
`in-range`	A field whose value lies within the allowed range (between the `min` and `max` attribute values)
`invalid`	A field whose value fails the validation test
`optional`	A field that is optional (not required) in the form
`out-of-range`	A field whose value lies outside the allowed range (outside the `min` and `max` attribute values)
`required`	A field that is required in the form
`valid`	A field whose value passes the validation test

Applying Inline Validation (continued 2)

- Example: To create styles for all of the checked option buttons in the form, apply the checked pseudo-class

```
input[type="radio"]:checked {
styles
}
```

- *styles* are the CSS styles applied to checked option buttons

Pseudo-Classes for Valid and Invalid Data

- The `valid` and `invalid` pseudo-classes are used to format controls based on whether their field values pass/fail a validation test
- Example: To display `input` elements containing valid data with a light green background, use the following style rule:

```
input:valid {
background-color: rgb(220, 255, 220);
}
```

Pseudo-Classes for Valid and Invalid Data (continued 1)

- Example: To display `input` elements containing invalid data with a light red background with focus, use the following style rule:

```
input:focus:invalid {
   background-color: rgb(255, 232,
233);
}
```

Pseudo-Classes for Valid and Invalid Data (continued 2)

Figure 7-57 Creating styles for valid and invalid field values

```
input:focus, select:focus, textarea:focus {
   background-color: rgb(220, 255, 220);
}

input#name:focus:valid,
input#zip:focus:valid,
input#phone:focus:valid,
input#mail:focus:valid {
   background: rgb(220, 255, 220) url(rb_valid.png) bottom right/contain no-repeat;
}

input#name:focus:invalid,
input#zip:focus:invalid,
input#phone:focus:invalid,
input#mail:focus:invalid {
   background: rgb(255, 232, 233) url(rb_invalid.png) bottom right/contain no-repeat;
}
```

style for valid data values in the selected fields that have the focus

display a green check mark image in the input box background

style for invalid data values in the selected fields that have the focus

display a red X image in the input box background