Tutorial 6
Creating a Web Form

HTML, CSS, and Dynamic HTML

5TH EDITION

Carey, P. (2013). New perspectives on HTML, CSS, and Dynamic HTML: Comprehensive (5th ed.). Australia ; United Kingdom: Course Technology Cengage Learning.

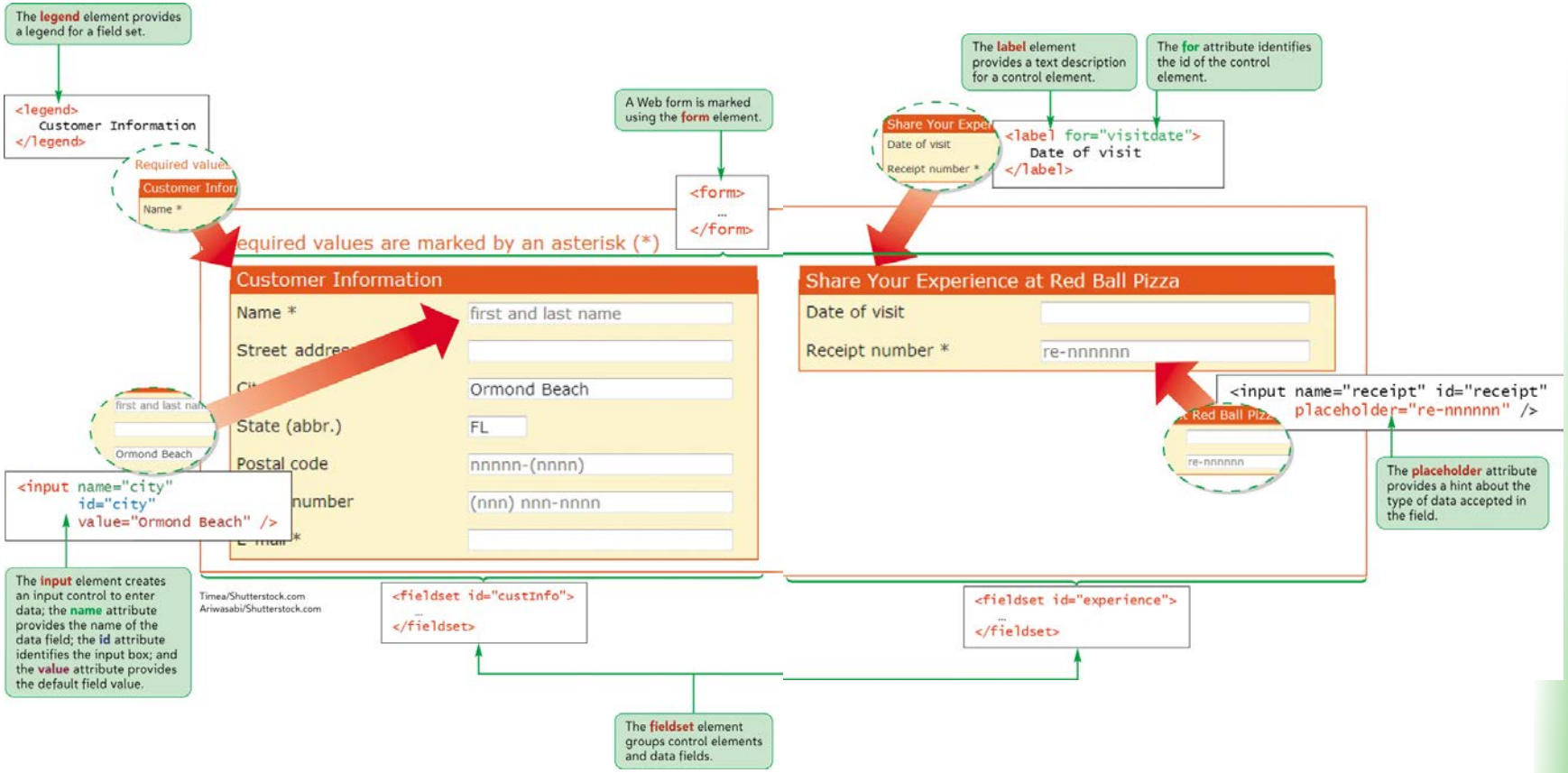
Objectives

- Explore how Web forms interact with Web servers
- Create form elements
- Create field sets and legends
- Create input boxes and form labels
- Create selection lists
- Creation option buttons
- Create text area boxes
- Create check boxes

Objectives

- Apply styles to Web forms
- Explore HTML5 data types
- Create spinners and range sliders
- Create form buttons
- Validate form data

Parts of a Web Form

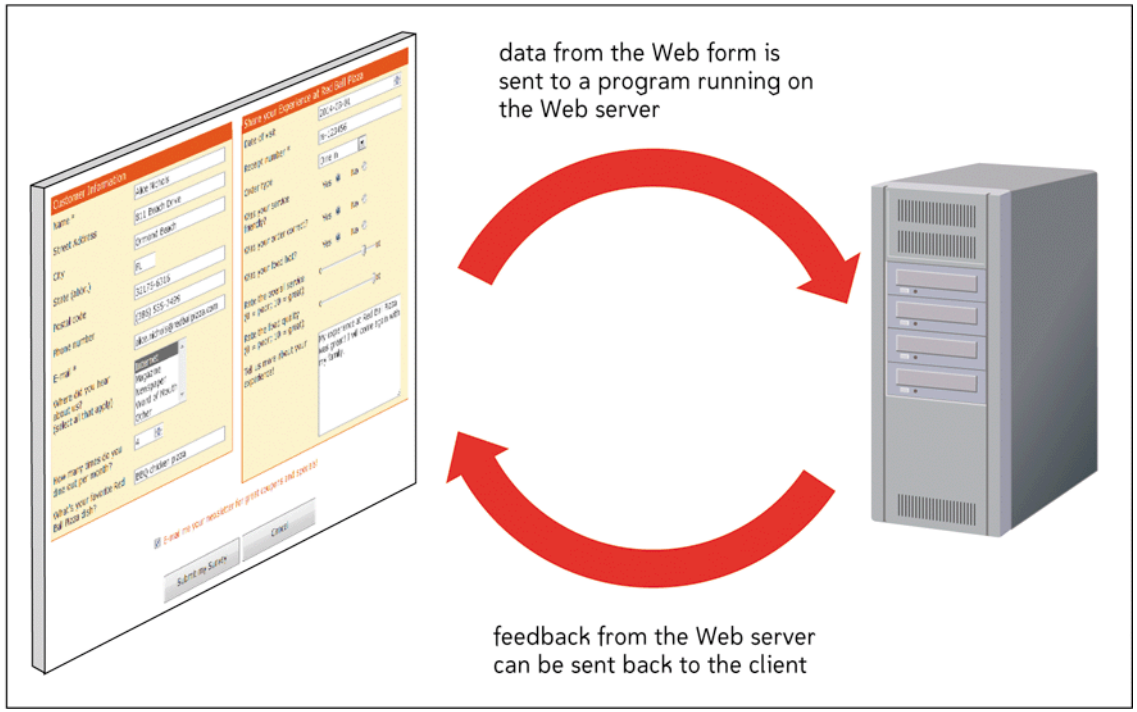


Introducing Web Forms

- Web forms collect information from Web site visitors.
- HTML supports the following control elements:
  - Input boxes
  - Option buttons
  - Selection lists
  - Check boxes
  - Text areas
  - Color pickers
  - Calendar pickers
  - Spin boxes
  - Sliders

Forms and Server-Based Programs

Figure 6-3 Interaction between a Web form and a Web server



Forms and Server-Based Programs

- Server-based programs are written in many languages
- The earliest and most commonly used are Common Gateway Interface (CGI) scripts that are written in Perl.
- Other popular languages include:
  - ASP/ASP.NET
  - ColdFusion
  - C/C++
  - Java
  - PHP
  - Python
  - Ruby

Creating a Web Form

- Forms are created using the form element, structured as follows:

```
<form attributes>
content
</form>
```

- Where *attributes* are the attributes that control how the form is processed and *content* is the content of the form.

Creating a Web Form

- Form attributes tell the browser the location of the server-based program to be applied to the form's data.
- Always specify an id or name for the form.
- Two attributes are available to identify the form: id and name.

Creating a Web Form

- The syntax of the id and name attributes are as follows:

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

- Where *id* is the id of the form and *name* is the name of the form.

Creating a Field Set

- HTML and XHTML allow you to organize a form into a group of fields called field sets.

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

where id identifies the field set and controls are the control elements associated with fields within the field set

Creating a Field Set

- To add a legend to a field set, add the following tag after the opening <fieldset> tag:

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

Where *text* is the text of the field set caption.

Creating Input Boxes

- The general syntax of input elements is as follows:

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

Where ***type*** specifies the type of input control, and the **name** and **id** attributes provide the control's name and id.

Creating Input Boxes

Figure 6-11 Input box data types

Type	Displays	General Appearance
`button`	A button that can be clicked to perform an action from a script	Run Program
`checkbox`	A check box that can be clicked by the user	
`file`	A Browse button to locate and select a file	C:\survey.htm Browse...
`hidden`	A hidden field, not viewable on the form	
`image`	An inline image that can be clicked to perform an action from a script	
`password`	An input box that hides text entered by the user	•••••••
`radio`	An option button that can be clicked by the user	
`reset`	A button that resets the form when clicked	Cancel Form
`submit`	A button that submits the form when clicked	Submit Form
`text`	An input box that displays text entered by the user	Alice Nichols

Adding Field Labels

- You can also expressly link text with a control element.
- The syntax for creating a form label is as follows:

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

Where id is the value of the id attribute for a field's control element, and label text is the text of the label.

Applying a Style Sheet to a Web Form

Figure 6-16 Styles for the form field sets

```
/* Field set styles */

fieldset {
   background-color: rgb(255, 246, 205);
   border: 1px solid rgb(233, 69, 0);
   float: left;
   margin: 10px 0px 10px 2.5%;
   width: 46%;
}
```

- sets the background color of each field set to a medium yellow
- adds a 1-pixel-wide solid red border to each field set
- floats each field set on the left
- sets the margins around the field sets
- sets the width of each field set to 46% of the total page width

Figure 6-17 Styles for the field set legends

```
legend {
   background-color: rgb(233, 69, 0);
   color: white;
   padding: 3px 0px;
   text-indent: 5px;
   width: 100%;
}
```

- displays the legends in white text on a red background
- sets the text indent and the padding space around the legend
- sets the width of each legend to 100% of the width of the field set

Applying a Style Sheet to a Web Form

Figure 6-18 Styles for the field labels

displays the label only when the left margin is clear

displays each label as a block floated on the left

sets the font size of each field label

sets the margin around each label

```
/* Label styles */

label {
   clear: left;
   display: block;
   float: left;
   font-size: 0.9em;
   margin: 7px 4% 7px 5px;
   width: 40%;
}
```

sets the label width to 40% of the width of the field set

Figure 6-19 Styles for input controls

displays each input control as a block, floated on the left

sets the font size of the text in the input control

sets the margin space around the input control

```
/* Input control styles */

input {
   display: block;
   float: left;
   font-size: 0.9em;
   margin: 7px 0px;
   width: 50%;
}
```

sets the width of each input control to 50% of the width of the field set

Defining Default Values and Placeholders

- To define the default value of a field, add the attribute

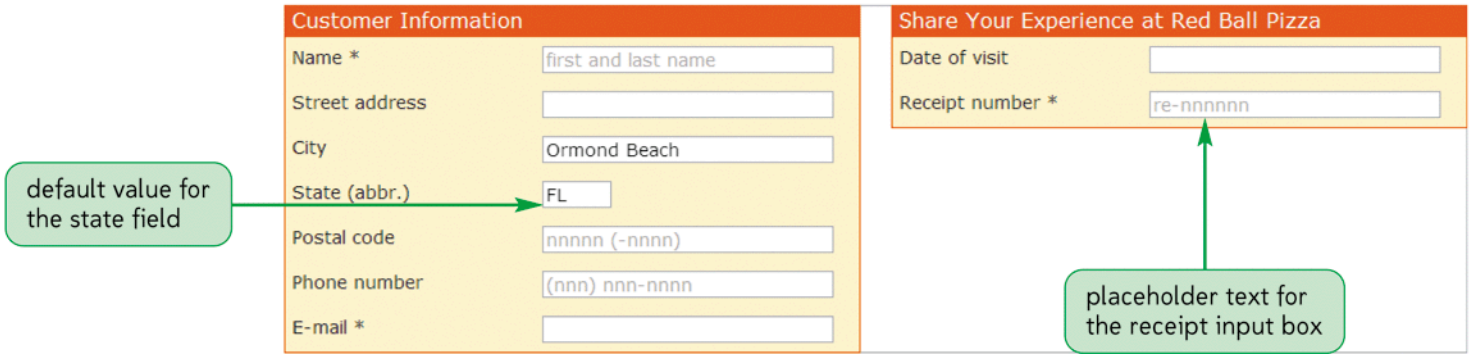
  `value="value"`

  to the control element, where *value* is the default value assumed by a browser unless a user enters a different value

- Starting with HTML5, you can also populate your input boxes with placeholders. A **placeholder** is a text string that appears within the control element and provides users with information about the kind of information accepted by the field

Defining Default Values and Placeholders

Figure 6-24 Viewing default values and placeholder text



Selection Lists and Option Buttons

```
<select name="ordertype">
   <option value="type1">Carry out</option>
   <option value="type2">Delivery</option>
   <option value="type3" selected>Dine in</option>
   <option value="type4">Take and bake</option>
</select>
```

The **select** element creates a selection list; the **option** elements provide the options in the list; **option text** is contained within the option element; and the default value is indicated by the **selected** attribute.

The **size** attribute specifies the number of options displayed in the selection list, and the **multiple** attribute allows a user to make multiple selections.

```
<select name="infoSrc" id="infoSrc"
        size="5" multiple="multiple" >
   <option value="internet">Internet</option>
…
</select>
```

Required values are marked by an asterisk (*)

Customer Information

Name * first and last name

Street address

Ormond Beach

FL

nnnnn-(nnnn)

Phon (nnn) nnn-nnnn

E-mail *

Where did you hear about us? (select all that apply)

Internet
Magazine
Newspaper
Word of Mouth
Other

Share Your Experience at Red Ball Pizza

Date of visit

Receipt number * re-nnnnnn

Order type Dine in

Was your service friendly? Yes No

Was your order correct? Yes No

Was your food hot? Yes No

Tell us more about your experience!

```
<input type="radio"
       name="orderCorrect"
       value="yes" />

<input type="radio"
       name="orderCorrect"
       value="no" />
```

The **radio** type displays a radio or option button; the **name** attribute defines the data field; and the **value** attribute provides the field value associated with the button.

The **textarea** element creates a text area box for multiple lines of text.

```
<textarea>…</textarea>
```

The **checkbox** type displays a check box control element in the Web form.

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

E-mail me your newsletter for great coupons and specials!

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

Timea/Shutterstock.com

Creating a Selection List

- A **selection list** is a list box from which a user selects a particular field value or set of field values.
  - Selection lists are useful when there are a fixed set of possible responses from the user.
- You can create a selection list using the <select> element.
- You can specify each individual selection item using the <option> element.

Creating a Selection List

- You can change the number of options displayed in the selection list by modifying the size attribute. The syntax is as follows:

```
<select size= “value”>… </select>
```

Where ***value*** is the number of items that the selection list displays in the form.

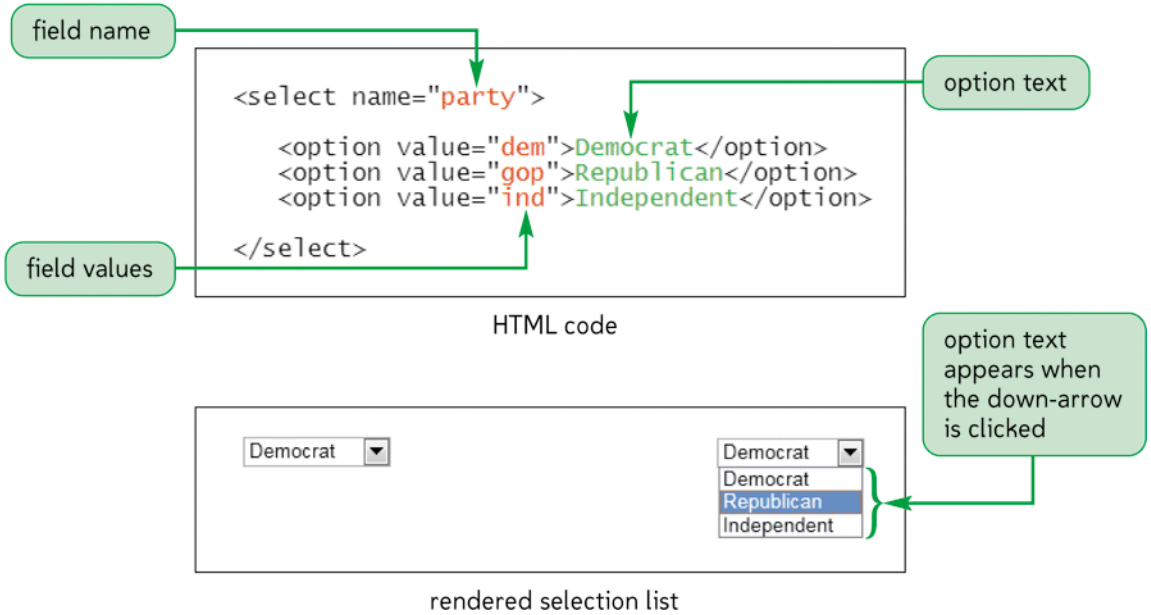
Creating a Selection List

- Add the multiple attribute to the select element to create multiple selections:

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

Creating a Selection List

Figure 6-25 Creating a selection list




```
<select name="party">

   <option value="dem">Democrat</option>
   <option value="gop">Republican</option>
   <option value="ind">Independent</option>

</select>
```

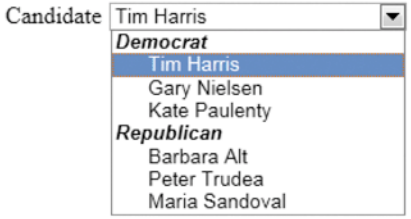
Grouping Selection Options

Figure 6-34 Organizing a selection list with option groups

HTML code

```
<label for="party">Candidate</label>
<select name="party">

   <optgroup label="Democrat">
      <option value="d1">Tim Harris</option>
      <option value="d2">Gary Nielsen</option>
      <option value="d3">Kate Paulenty</option>
   </optgroup>

   <optgroup label="Republican">
      <option value="r1">Barbara Alt</option>
      <option value="r2">Peter Trudea</option>
      <option value="r3">Maria Sandoval</option>
   </optgroup>

</select>
```

rendered selection list



Creating Option Buttons

- **Option buttons**, or **radio buttons** allow users to make selections.
  - Unlike selection lists, option buttons only allow the user to select one option at a time.

Figure 6-35 Creating a group of option buttons

HTML code

```
<fieldset>
   <legend>Party Affiliation<legend>

   <label for="demOption">Democrat</label>
   <input type="radio" name="party" id="demOption" value="dem" />

   <label for="repOption">Republican</label>
   <input type="radio" name="party" id="repOption" value="rep" />

   <label for="indOption">Independent</label>
   <input type="radio" name="party" id="indOption" value="ind" />

</fieldset>
```

rendered option buttons

Party Affiliation
Democrat ◯ Republican ◯ Independent ◯

Creating a Group of Option Buttons

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

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

  <input type="radio" name="*name" value="value2" />*

  <input type="radio" name="*name" value="value3" />*

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

- To specify the default option, add the checked attribute to the input element as follows:

  checked=”checked”

Creating a Text Area Box

- Text area boxes allow users to enter comments.
- An input box would be too small to accommodate the length of text for this use.

Creating a Text Area Box

- To create a text area box, use the textarea element:

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

Where the rows and cols attributes define the dimensions of the input box and the rows attribute indicates the number of lines in the input box.

Creating a Text Area Box

- As you type text into a text area box, the text automatically wraps to a new line as it extends beyond the box's width
- You can determine whether the locations of line wrapping are included in the field value by using the wrap attribute
  - Hard
  - Soft

Creating Check Boxes

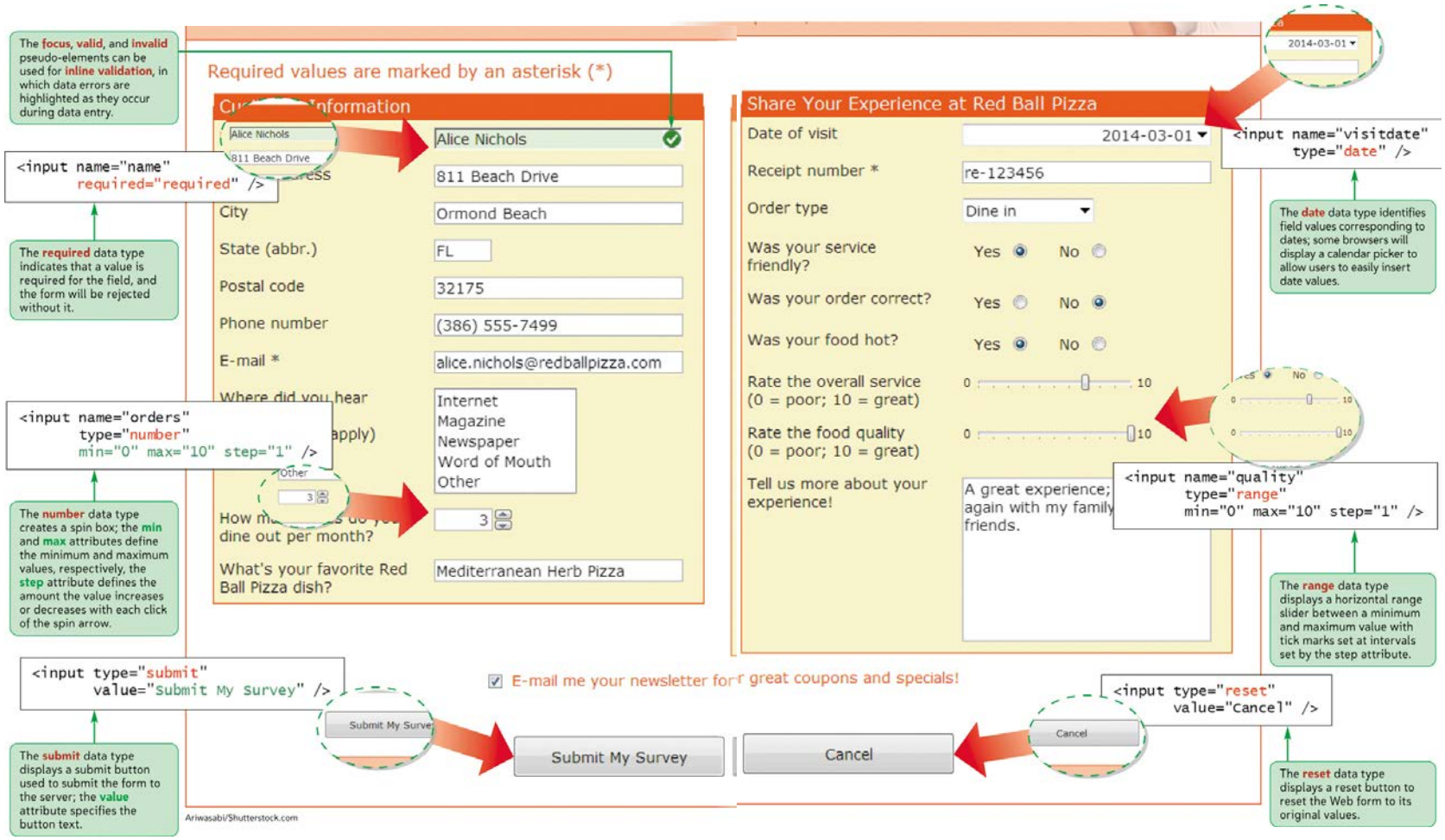
- To create a check box, use:

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

- Where the name attribute identifies the check box controls and the value attribute specifies the value sent to the server if the check box is selected.
- To specify that a check box be selected by default, use the checked attribute as follows:

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

HTML5 Data Types



Exploring HTML5 Data Types

Figure 6-48 HTML5 data types

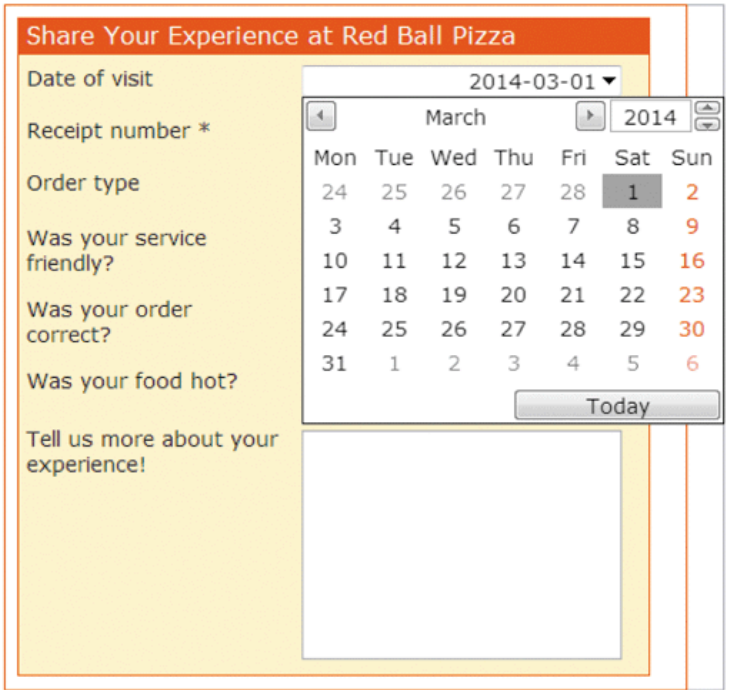
Type	Description	General Appearance
`color`	An RGB color value that can be selected from a color picker dialog box	
`date`	A date (year, month, day) with no specified time zone	
`datetime`	A date and time (year, month, day, hour, minute, second, fraction of a second) with the time zone set to Coordinated Universal Time (UTC)	
`datetime-local`	A date and time (year, month, day, hour, minute, second, fraction of a second) with no specified time zone	
`email`	An e-mail address or list of e-mail addresses	nichols@redballpizza.com
`month`	A date consisting of a year and a month	
`number`	A numeric value	5
`range`	A numeric value selected from a defined range of values	
`search`	A text string usually used for performing searches	local pizza
`tel`	A telephone number	(365) 555 - 7499
`time`	A time value (hour, minute, seconds, fractional seconds)	01:45
`url`	A URL of a Web site or Internet resource	http://www.redballpizza.com
`week`	A date consisting of a year number and a week number	

Exploring HTML5 Data Types

- The email, tel, and url data types are used for storing e-mail addresses, telephone numbers, and Web addresses, respectively
- For browsers that support the date type, this will bring up a calendar widget from which users can select a date

Exploring HTML5 Data Types

Figure 6-52 Calendar widget in the Opera browser



Creating Spinner Controls and Range Sliders

- To create a spinner control for numeric data, enter the input element

```
<input name="name" type="number" value="value"
step="value" min="value" max="value" />
```

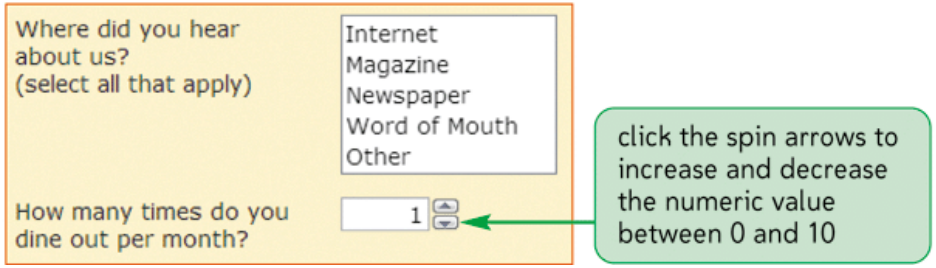
where the value attribute provides the default field value, the step attribute indicates the amount by which the field value changes when a user clicks the spin arrow, the min attribute defines the minimum possible value, and the max attribute defines the maximum possible value of the field

- To create a range slider control for numeric data, use the following input element:

```
<input name="name" type="range" value="value"
step="value" min="value" max="value" />
```

Creating Spinner Controls and Range Sliders

Figure 6-55 Number spin box in the Opera browser



Specifying a Numeric Range with the range Data Type

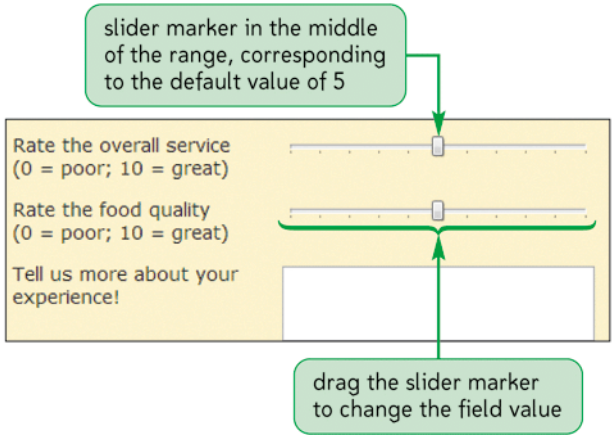
Figure 6-56 Adding input elements with the range data type

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

<label>Rate the food quality<br />
       (0 = poor; 10 = great)</label>
<input name="quality" id="quality" type="range" value="5"
       min="0" max="10" step="1" />

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

range data type

quality values range from 0 to 10 in steps of 1

Figure 6-57 Range slider in the Opera browser



Creating and Applying a Data List

- To create a data list of possible values, enter the HTML code

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

where the value attribute provides the text of the possible values in the data list.

Creating and Applying a Data List

- To reference the data list from an input control, add the list attribute

  <input name="*name*" list="*id*" />

where *id* references the id of the data list structure.

Creating Form Buttons

- Form buttons are a type of control element that performs an action.
- Types of buttons:
  - Command button
  - Submit button
  - Reset button

Creating a Command button

- Command buttons are created using the <input> tag:

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

- Submit buttons submit forms to the server for processing when clicked. Syntax is as follows:

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

- Reset buttons reset forms to their original (default) values. Syntax is as follows:

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

Designing a Custom Button

- Use the button element for greater artistic control over the appearance of a button.

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

where the type attribute specifies the button type (submit, reset, or button—for creating a command button) and *content* is page elements displayed within the button

Designing a Custom Button

Figure 6-66 Creating a custom button

HTML code

```
<button type="button">

   <img src="home.png" />
   Return to the <br /> Home Page

</button>
```

custom button

Validating a Web Form

- Data values often need to be tested or **validated** before they can be used.
  - **Server-side validation**
  - **Client-side validation**

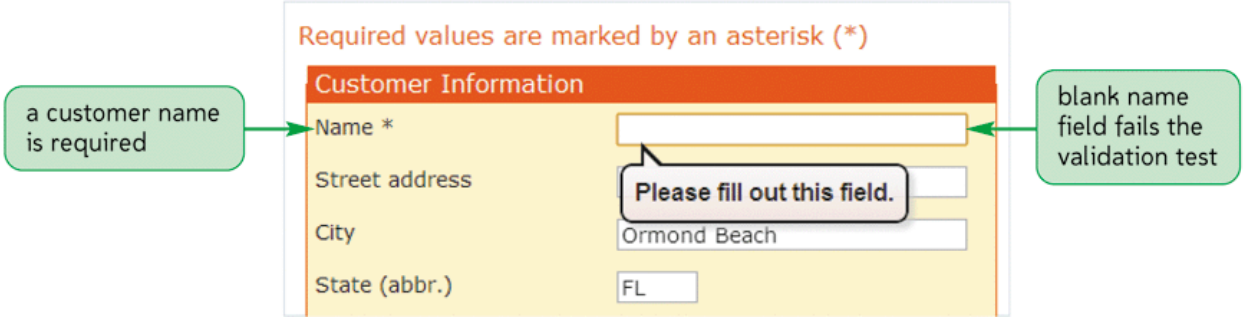
Validating Field Values

- To indicate that a field is required, add the required=”required” attribute to the control element
- To validate an e-mail address, set the data type to email. To validate a Web address, set the data type to url
- To validate that a text input box follows a character pattern, add the attribute `pattern=”`*`regex`*`”`

where *regex* is a regular expression that defines the character pattern.

Validating Field Values

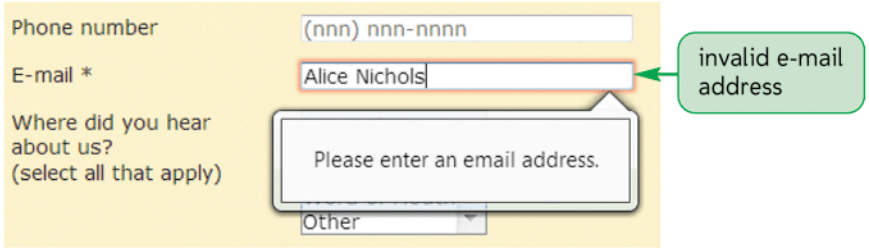
Figure 6-68 Data validation error message in Google Chrome



Validating Based on Data Type

- The new data types supported by HTML5 also can be used for data validation.
  - A data field with the number data type will be rejected if non-numeric data is entered.
  - Similarly, fields marked using the email and url fields will be rejected if a user provides an invalid e-mail address or Web site URL

Figure 6-69 Entering an invalid e-mail address



Testing for a Valid Pattern

Figure 6-70 Specifying character patterns with regular expressions

```
<label for="zip">Postal code</label>
<input name="zip" id="zip"
       placeholder="nnnnn (-nnnn)"
       pattern="^\d{5}(\-\d{4})?$" />

<label for="phone">Phone number</label>
<input name="phone" id="phone" type="text"
       placeholder="(nnn) nnn-nnnn"
       pattern="^\d{10}$|^(\(\d{3}\)\s*)?\d{3}[\s-]?\d{4}$" />
```

regular expression for a five- or nine-digit postal code

data type changed to text

regular expression for a phone number

```
<label for="receipt">Receipt number *</label>
<input name="receipt" id="receipt"
       placeholder="re-nnnnnn"
       required="required"
       pattern="^re\-\d{6}$" />
```

regular expression for a Red Ball Pizza receipt number

Applying Inline Validation

- One disadvantage with the current validation checks is that they all occur *after* a user has completed and submitted the form
- The technique of immediate data validation and reporting of errors is known as **inline validation**
- One way of integrating inline validation into a Web form is to create style rules that change the appearance of each control element based on the validity of the data it contains
  - This can be done using some of the CSS3 pseudo-classes

Applying Inline Validation

Figure 6-72 Pseudo-classes for Web form controls

Pseudo-Class	Matches
`checked`	Check boxes or options that are checked
`default`	The default user control element
`disabled`	Control elements that are disabled
`enabled`	Control elements that are enabled
`focus`	Control elements that have the focus (are actively selected) in the form
`indeterminate`	Check boxes or option buttons whose toggle states (checked or unchecked) cannot be determined
`in-range`	Control elements whose values are within each field's range of values (between the `min` and `max` attribute values)
`invalid`	Control elements whose values fail validation tests
`optional`	Control elements that are optional (not required) in the Web form
`out-of-range`	Control elements whose values are outside each field's range of values (outside of the `min` and `max` attribute values)
`required`	Control elements that are required in the Web form
`valid`	Control elements whose values pass validation tests

Applying Inline Validation

Figure 6-73 Style rule for elements that have the focus

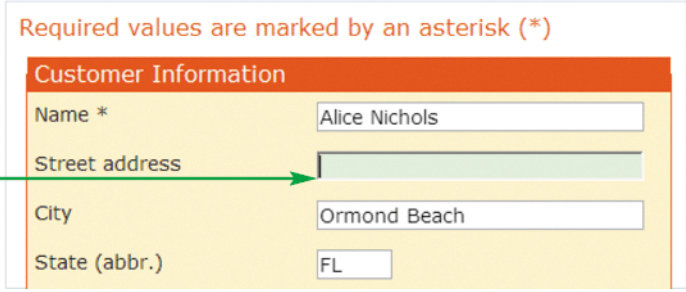
selector for input, select, and textarea elements that have the focus

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

changes the background color to light green

Figure 6-74 Changed background color for element that has the focus

active input element has the focus with a light green background



Pseudo-Classes for Valid and Invalid Data

Figure 6-75 Style rules for valid and invalid field values

```
/* Validation styles */

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

input:focus:valid {
   background: rgb(220, 255, 220) url(go.png) bottom right no-repeat;

   -o-background-size: contain;
   -moz-background-size: contain;
   -webkit-background-size: contain;
   background-size: contain;
}

input:focus:invalid {
   background: rgb(255, 232, 233) url(stop.png) bottom right no-repeat;

   -o-background-size: contain;
   -moz-background-size: contain;
   -webkit-background-size: contain;
   background-size: contain;
}
```

style rule for valid data values

displays the *go.png* image file in the bottom-right corner of the input box on a light green background

resizes the image so that it is contained within the input box

style rule for invalid data values

displays the *stop.png* image file in the bottom-right corner of the input box on a light red background

resizes the image so that it is contained within the input box

Pseudo-Classes for Valid and Invalid Data

Figure 6-76 Inline validation on the postal code