

1 ☐ **WEB INPUT FORMS**

HTML5 and CSS3

2 ☐ **USER INPUT FORMS—PAGE 1**

- Input forms used on web pages to get data from users
 - Information might include reader feedback (comments, complaints, etc.), on-line orders (sales/e-commerce), or simple data entry into databases.

4 ☐ **USER INPUT FORMS—PAGE 2**

- There are two parts to creating and using forms:
 - Create the form itself
 - Process the results (requires knowing a language such as JavaScript or Perl that can manipulate the data either on the client or on the web server)

5 ☐ **FORM DIALOG ELEMENTS**

- There are three basic classifications of form dialog elements:
 - The input tag which includes several types of form elements including:
 - One-line text box, password box, radio button, check box and push buttons
 - HTML5 introduces additional input elements including color pickers, calendar pickers, spinners and sliders
 - The select tag for drop-down menus and list boxes
 - The textarea tag which is a multi-line scrolling text box

9 ☐ **FORM CONTROLS—PAGE 1**

- HTML supports the following control elements:
 - Input boxes—for text and numerical entries
 - Option buttons (also called radio buttons)—for selecting a single option from a predefined list
 - Selection lists—for long lists of options, usually appearing in a drop-down list box
 - Check boxes—for questions limited to true/false or yes/no responses
 - Text areas—for extended entries that can include several lines of text

10 ☐ **FORM CONTROLS—PAGE 2**

- HTML5 introduced the following control elements:
 - Color pickers—to choose colors from an array of color values
 - Calendar pickers—to select dates and times from a calendar or clock
 - Spin boxes—for specifying numeric values from sets of numbers
 - Sliders—for selecting numeric values from ranges of possible values

11 ☐ **HTML FORM SYNTAX**

- The basic form creates a block using the form tags
- A number of form elements may be contained inside this block
- Format:

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

12 ☐ IDENTIFYING THE FORM—PAGE 1

- There are two attributes in the form tag that are used for identification:
- The id attribute differentiates the form from other elements on the web page
- The name attribute differentiates one form from another when there are multiple forms

13 ☐ IDENTIFYING THE FORM—PAGE 2

- Format:

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

- The id and name attribute values often are the same

14 ☐ IDENTIFYING THE FORM—PAGE 3

- Example:

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

16 ☐ INTERACTING WITH THE WEB SERVER—PAGE 1

- Other form attributes specify where and how to send the form data
- The action attribute specifies the location and filename of the program that will process the file

17 ☐ INTERACTING WITH THE WEB SERVER—PAGE 2

- Other form attributes specify where and how to send the form data (*con.*)
- The method attribute specifies how form data will be passed to web server
 - The value get means the data will be appended to the end of the url specified in the action attribute
 - The value post sends the data in a separate data stream

18 ☐ INTERACTING WITH THE WEB SERVER—PAGE 3

- Format:

```
<form action="url" method="get | post">
  form content
</form>
```

- Example:

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

</form>

19 ENCODING THE FORM DATA

- The enctype property for the form tag specifies how the data is encoded when it is sent to the web server
- The most common types are:
 - application/x-www-form-urlencoded—transferred as a long string; white space replaced by “+” and nontext characters replaced by hexadecimal code values
 - multipart/form-data—spaces and nontext characters preserved; data elements separated using delimiter lines; method="post" required
 - text/plain—no encoding of spaces and nontext characters; most often used when action="mailto: ..."

21 CREATING A FIELD SET—PAGE 1

- The fieldset block around controls (elements) on a form places a box (actually a frame) around those controls
- Format:


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

22 CREATING A FIELD SET—PAGE 2

- Example:


```
<fieldset id="custInfo">
    Name *
    <input name="custname" id="custname">
    Street address
    <input name="street" id="street" />
    City
    <input name="city" id="city" />
    ...
    E-mail *
    <input name="email" id="email">
</fieldset>
```

24 ADDING A LEGEND TO A FIELD SET—PAGE 1

- Within the fieldset block, legend tags may be used to place a label on a border of the frame
- The align attribute of the legend tag allows the label to appear at the top or bottom of the frame, and/or the left or right of it (top and left are the defaults)
- Format:


```
<fieldset id="id">
    <legend align="top | bottom | left | right">text</legend>
```

```

        controls
    </fieldset>

```

25 ADDING A LEGEND TO A FIELD SET—PAGE 2

- Example:

```

<fieldset id="custInfo">
    <legend>Customer Information</legend>
    Name *
    <input name="custname" id="custname">
    Street address
    <input name="street" id="street" />
    City
    <input name="city" id="city" />
    ...
    E-mail *
    <input name="email" id="email">
</fieldset>

```

27 INPUT BOXES—PAGE 1

- Input “boxes” are created using the single-sided HTML input tag and include the following:
 - button—can be clicked to perform an action from a script
 - checkbox—a check box that can be clicked on and off
 - file—a “browse” button to locate and select a file
 - hidden—not viewable on the form
 - image—can be clicked to perform an action from a script
 - password—input box that hides text entered by user

28 INPUT BOXES—PAGE 2

- Input “boxes” include the following (*con.*):
 - radio—an option button that can be clicked by user
 - reset—button that resets the form
 - submit—submits form data to web server when clicked
 - text—an input box that displays the text entered by user (default type)

29 INPUT BOXES—PAGE 3

- Format for creating an input box:


```
<input type="type" attributes />
```

 - The tag is one-sided with an ending slash (/)
- Example:


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

 - Creates a one-line text box

- The name attribute is used for storing value passed to processor
- The id attribute is used by the processor for CSS formatting

30 THE ONE-LINE TEXT BOX—PAGE 1

- The one-line text box is created by using the input tag with the type attribute assigned the value "text"
- Format:
`<input type="text" attributes />`

31 THE ONE-LINE TEXT BOX—PAGE 2

- Examples:
`<input type="text" name="custname" id="custname" />`
`<input name="custname" id="custname" />`
 - The entry type="text" is not required for a text box as per the second example since it is the *default* type

33 ADDING FIELD LABELS—PAGE 1

- A "field label" is added using the label tag and provides the following functionality:
 - A descriptive text for a form element (rather than just type text next to a control)
 - If user clicks on the label, the form element associated with it receives the focus assuming the for attribute is set to a specific control

34 ADDING FIELD LABELS—PAGE 2

- Format:
`<label for="id">label text</label>`
 - The *id* is the name of the control with which the label is associated
- Example:
`<label for="custname">Name *</label>`
`<input type="text" name="custname" id="custname" />`

35 ADDING FIELD LABELS—PAGE 3

- Alternate format (wrap the element inside the <label> block):
`<label>label text`
`<formElement attributes>`
`</label>`
 - The for attribute is not needed if form element is placed inside label block
- Alternate example:
`<label>Student name:`
`<input type="text" name="custname" id="custname" />`
`</label>`

37 DEFINING DEFAULT VALUES—PAGE 1

- The value attribute provides an initial value for a text box control that is stored and displayed in the field when the form first loads

- Current “value” in each text box as assigned to the name (like a variable) when it passed to the server when the form is submitted

38 DEFINING DEFAULT VALUES—PAGE 2

- Format:

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

 - The *value* is assigned to the *name* attribute (variable)
- Example:

```
<input type="text" name="state" id="state" value="FL" />
```

40 DEFINING PLACEHOLDERS

- The placeholder attribute displays a descriptive string (greyed out) to the user about the type of information is accepted by the field
- Format:

```
<input type="text" name="name" id="id" placeholder="text" />
```
- Example:

```
<input type="text" name="zip" id="zip" placeholder="nnnnn (-nnnn)" />
```

42 CREATING A SELECTION LIST—PAGE 1

- The select block creates a scrollable menu (drop-down list or list box) used to select from a list of choices
- List of choices are inserted into the menu with a series of option blocks

43 CREATING A SELECTION LIST—PAGE 2

- Format:

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

 - The *value* of the option selected is assigned to the *name* (variable) property in the select when the form is submitted
 - The *text* are values displayed in the selection list

44 CREATING A SELECTION LIST—PAGE 3

- Example:

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

46 SETTING A DEFAULT VALUE FOR A SELECTION LIST—PAGE 1

- The selected attribute is used in the option tag to make that option the default entry from the list when the web page loads
- Only one option may be selected as the default

47 ☐ **SETTING A DEFAULT VALUE FOR A SELECTION LIST—PAGE 2**

- Format:

```
<option selected="selected" value="value">text</option>
```
- Most browsers will correctly interpret the selected attribute even if ="selected" is not specified:

```
<option selected value="value">text</option>
```

48 ☐ **SETTING A DEFAULT VALUE FOR A SELECTION LIST—PAGE 3**

- Example:

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

50 ☐ **SETTING THE SIZE OF THE SELECTION LIST—PAGE 1**

- By default, select lists are "drop-down lists" that display only currently selected option
- The size attribute if set to a value *greater than "1"* creates a "list box" so that a number of options are visible at the same time
- A vertical scroll bar appears automatically if size is smaller than the number of options in the list

51 ☐ **SETTING THE SIZE OF THE SELECTION LIST—PAGE 2**

- Format:

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

52 ☐ **SETTING THE SIZE OF THE SELECTION LIST—PAGE 3**

- Example:

```
<select name="ordertype" id="ordertype" size="5">
  <option value="internet">Internet</option>
  <option value="mag">Magazine</option>
  <option value="news">Newspaper</option>
  <option value="word">Word of Mouth</option>
  <option value="other">Other</option>
</select>
```

54 ☐ ALLOWING FOR MULTIPLE SELECTIONS—PAGE 1

- The multiple property allows the user to select more than one option from the list
- To do this requires use of the <Ctrl> and/or the <Shift> key on the keyboard while clicking the item(s) (or the user may click and drag)
- If this property is turned “on” the control always will appear as a list box

55 ☐ ALLOWING FOR MULTIPLE SELECTIONS—PAGE 2

- Format:
`<select multiple="multiple" attributes>`
...
`</select>`
- Most browsers will correctly interpret the multiple attribute even if “multiple” is not specified:
`<select multiple attributes>`
...
`</select>`

56 ☐ ALLOWING FOR MULTIPLE SELECTIONS—PAGE 3

- Example:
`<select name="ordertype" id="ordertype" multiple="multiple" >`
 `<option value="internet">Internet</option>`
 `<option value="mag">Magazine</option>`
 `<option value="news">Newspaper</option>`
 `<option value="word">Word of Mouth</option>`
 `<option value="other">Other</option>`
`</select>`

58 ☐ GROUPING SELECTION OPTIONS—PAGE 1

- For longer lists options can be placed into groups using an optgroup block
- Format:
`<select>`
 `<optgroup label="label1">`
 `<option value="value">text</option>`
 ...
`</optgroup>`
 `<optgroup label="label2">`
 `<option value="value">text</option>`
 ...
`</optgroup>`
`</select>`

59 ☐ GROUPING SELECTION OPTIONS—PAGE 2

- Example:

```
<select name="vote" id="vote">
  <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>
```

60 CREATING OPTION BUTTONS—PAGE 1

- The option button always is created as a group of buttons to limit a user's choices to one of a series of related choices
- Also known as radio buttons
- An identical "group" name property value should be given to all option buttons in the group
- Clicking and selecting one option button turns off any other button in group

61 CREATING OPTION BUTTONS—PAGE 2

- Option buttons are created using the input tag with the type="radio"
- Format:


```
<input type="radio" name="name" value="value1" />
<input type="radio" name="name" value="value2" />
```

 - value property values are assigned to the name (variable)

62 CREATING OPTION BUTTONS—PAGE 3

- Examples:


```
<input type="radio" name="serviceFriendly" id="sYes" value="yes" />
<input type="radio" name="serviceFriendly" id="sNo" value="no" />
```

 - name property is the *same* for all buttons in the *group*

63 SETTING DEFAULT VALUES FOR OPTION BUTTONS—PAGE 1

- The checked attribute is used in an option button group to make that option the default entry from the group when the web page loads
- Only one option button may be checked as the default

64 SETTING DEFAULT VALUES FOR OPTION BUTTONS—PAGE 2

- Format:


```
<input type="radio" name="name" checked="checked" value="value" attributes
```

`</>`

- Most browsers will correctly interpret the selected attribute even if `= "checked"` is not specified:

```
<input type="radio" name="name" checked value="value" attributes />
```

65 **SETTING DEFAULT VALUES FOR OPTION BUTTONS—PAGE 3**

- Examples:

```
<input type="radio" name="serviceFriendly" id="sYes" value="yes" />
```

```
<input type="radio" name="serviceFriendly" id="sNo" checked="checked" value="no" />
```

67 **CREATING A TEXT AREA BOX—PAGE 1**

- The textarea block creates a multiple-line text
- Useful for allowing users to enter several lines of information such as narrative comments and/or problems
- When more text is typed than will fit, scroll bars appear automatically

68 **CREATING A TEXT AREA BOX—PAGE 2**

- The size can be set by modifying attributes for both width in characters (the `cols` attribute) and number of lines (the `rows` attribute)
- Size also can be configured using CSS
- Default text for textarea may be typed between the opening `<textarea>` and closing `</textarea>` tags

69 **CREATING A TEXT AREA BOX—PAGE 3**

- Format:

```
<textarea rows="value" cols="value" wrap="hard | soft"></textarea>
```

- The wrap *type* values:
 - hard—text is wrapped when submitted to server
 - soft—text is not wrapped when submitted to server

70 **CREATING A TEXT AREA BOX—PAGE 4**

- Example:

```
<textarea name="comments" id="comments" rows="5" cols="60"></textarea>
```

72 **CREATING CHECK BOXES—PAGE 1**

- The check box is an on/off button that can be selected and unselected
- The first time it is clicked, a check mark (✓) is displayed in the box, and the next time it is clicked, the check mark is turned off
- Unlike option buttons, check boxes are not grouped
 - Therefore more than one may be checked (clicked “on”) at the same time

73 **CREATING CHECK BOXES—PAGE 2**

- Check boxes are created using the input tag with the `type="checkbox"`
- Format:

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

- The value property values are assigned to the name (variable) when the form is submitted to server

74 CREATING CHECK BOXES—PAGE 3

- Example:

`<input type="checkbox" name="news cb" />`

75 SETTING A CHECK BOX “ON” WHEN FORM LOADS—PAGE 1

- The checked attribute is used with a check box to set it “on” when the web page loads

76 SETTING A CHECK BOX “ON” WHEN FORM LOADS—PAGE 2

- Format:

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

- Most browsers will correctly interpret the checked attribute even if “checked” is not specified:

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

77 SETTING A CHECK BOX “ON” WHEN FORM LOADS—PAGE 3

- Example:

`<input type="checkbox" checked="checked" name="news cb" />`

79 THE EMAIL DATA TYPE—PAGE 1

- An e-mail text box is created in HTML5 by using the input tag with the type = “email”
- Browser will validate that user enters a valid e-mail address
- Not supported by all browsers

80 THE EMAIL DATA TYPE—PAGE 2

- Format:

`<input type="email" attributes />`

- Example:

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

81 THE TEL DATA TYPE—PAGE 1

- A telephone number text box is created in HTML5 by using the input tag with the type = “tel”
- Browser will validate that user enters a valid phone number
- Not supported by all browsers

82 THE TEL DATA TYPE—PAGE 2

- Format:

`<input type="tel" attributes />`

- Example:

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

83 **THE URL DATA TYPE—PAGE 1**

- A web address text box is created in HTML5 by using the input tag with the type = "url"
- Browser will validate that user enters a valid URL
- Not supported by all browsers

84 **THE URL DATA TYPE—PAGE 2**

- Format:

```
<input type="url" attributes />
```
- Example:

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

86 **SPECIFYING DATE AND TIME—PAGE 1**

- A date/time text box is created by in HTML5 using the input tag with the type = "date"
- Some browsers will display a date dialog window ("picker") and validate that user enters a valid date
- Not supported by all browsers

87 **SPECIFYING DATE AND TIME—PAGE 2**

- Other date/time values for the type attribute of an input tag include:
 - datetime—to select both date and time with time zone option from a calendar and a list of hours, minutes, and AM or PM
 - datetime-local—to select both date and time without time zone option from a calendar and a list of hours, minutes, and AM or PM
 - month—to select month from a list of months
 - time—to select time from a list of hours, minutes, and AM or PM
 - week—to select a week number from a list of weeks of the year

88 **SPECIFYING DATE AND TIME—PAGE 3**

- Format:

```
<input type="date" attributes />
```
- Example:

```
<input type="date" "visitDate" id="visitDate" />
```

90 **THE NUMBER DATA TYPE—PAGE 1**

- A "spinner" control is created by in HTML5 using the input tag with type = "number"
- User clicks "up" and "down" arrows to advance numbers
- Not supported by all browsers

91 **THE NUMBER DATA TYPE—PAGE 2**

- Format:

```
<input type="number" value="defaultValue" min="minimumValue"
      max="maximumValue" step="stepValue" attributes />
```

- value—default value when form loads
- min—minimum value the spinner can represent
- max—maximum value the spinner can represent
- step—the amount by which the spinner advances at each click

92 THE NUMBER DATA TYPE—PAGE 3

- Example:

```
<input type="number" name="ordersPerMonth" id="ordersPerMonth" value="1"
      min="0" max="10" step="1" />
```

94 CREATING A RANGE ELEMENT—PAGE 1

- A range (horizontal slider) control is created in HTML5 by using the input tag with the type = "range"
- User drags the slider marker "left" and "right" to advance the values
- Not supported by all browsers

95 CREATING A RANGE ELEMENT—PAGE 2

- Format:

```
<input type="range" value="defaultValue" min="minimumValue"
      max="maximumValue" step="stepValue" attributes />
```

- value—default value when form loads
- min—minimum value the spinner can represent
- max—maximum value the spinner can represent
- step—amount by which the slider advances when dragged by the mouse

96 CREATING A RANGE ELEMENT—PAGE 3

- Example:

```
<input name="service" id="service" type="range" value="1" min="0" max="10"
      step="1" />
```

98 CREATING FORM BUTTONS—PAGE 1

- A button control lets the user do one of three things:
 - Submits the form to a file for processing by a file handler script located on the server
 - Resets (clears) the form
 - Initiates execution of a script located in the HTML web page written in JavaScript or some other web programming language

99 CREATING FORM BUTTONS—PAGE 2

- The button is created by using input tag with the type attribute assigned one of three values depending upon which type of button is needed
- Format:

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

- submit—sends form to a file for processing by form handler on the server
- reset—clears the form
- button—executes a local script within the web page

100 CREATING FORM BUTTONS—PAGE 3

- Format:

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

- The value attribute *text* is displayed as the button label; defaults are:
 - “Submit Query” for the submit button
 - “Reset” for the reset button
 - Default button label is blank and always should include a value attribute
- The onclick attribute only is used for a command button to run a client-side *script* statement

101 CREATING FORM BUTTONS—PAGE 4

- Examples:

`<input type="submit" value="Submit My Form" />`

`<input type="reset" value="Reset Form" />`

103 INDICATING REQUIRED VALUES—PAGE 1

- Use the required attribute with an input textbox to specify that the field may not be blank when submitted to the server
- The attribute works with input types text, date, search, url, tel, email, password, number, checkbox, radio and file

104 INDICATING REQUIRED VALUES—PAGE 2

- Format:

`<input type="type" required attributes />`

- Example:

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

106 TESTING FOR A VALID PATTERN—PAGE 1

- Use the pattern attribute with an input textbox to specify a “regular expression” upon which the element’s value is validated before the form is submitted to the server
- The attribute works with input types text, date, search, url, tel, email and password

107 TESTING FOR A VALID PATTERN—PAGE 2

- Format:

`<input type="type" pattern="regex" attributes />`

- The *regex* value is a “regular expression” that specifies the validation to be implemented on the user input
- Examples of regular expressions can be found on many websites

- Example:
`<input name="zip" id="zip" pattern="\d{5}([\-]\d{4})?" />`
- This pattern tests for a valid zip code

109 USING THE FOCUS PSEUDO-CLASS—PAGE 1

- The CSS focus pseudo-class can be implemented to highlight in some way an HTML element that currently has the focus
- It is triggered when the user clicks mouse on the element to enter data

110 USING THE FOCUS PSEUDO-CLASS—PAGE 2

- Format:
selector: focus
{
 property1: value1; ...
}
- Example:
`input:focus`
{
 background-color: yellow;
}

112 PSEUDO-CLASSES FOR VALID AND INVALID DATA—PAGE 1

- The following CSS pseudo-classes can provide formatting for valid and invalid data input relative to the required and pattern attributes as the data is entered:
 - required—input was required
 - valid—input is valid
 - invalid—input is invalid
- The valid and invalid pseudo-classes also apply validation to the tel, email and url attributes

113 PSEUDO-CLASSES FOR VALID AND INVALID DATA—PAGE 2

- Format:
selector: pseudoClass
{
 property1: value1; ...
}

114 PSEUDO-CLASSES FOR VALID AND INVALID DATA—PAGE 3

- Examples:
`input: required`
{
 border-color: red;
}

```
input:invalid
{
    color: red; font-weight: bold;
}
```

116 CREATING A DATA LIST—PAGE 1

- A data list control is created in HTML5 by using the input tag with a list attribute that names a datalist block that contains “suggested” options
- The datalist block is similar to a select list, but the list only *suggests* possible responses and users may enter anything that they want

117 CREATING A DATA LIST—PAGE 2

- Format (the option tags may be single-sided):

```
<input list="dataListID" attributes>
<datalist id="id" attributes>
    <option value="value1" />
    <option value="value2" />
    ...
</datalist>
```

 - The values of the options are text that appears in the list and the option selected is assigned to name property when the form is submitted

118 CREATING A DATA LIST—PAGE 3

- Example:

```
<input name="favDish" id="favDish" list="dishType" />
<datalist id="dishType">
    <option value="Antipasto Pizza" />
    <option value="Big Kahuna Pizza" />
    <option value="BBQ Chicken Pizza" />
    <option value="Mediterranean Herb Pizza" />
    <option value="Pasta Rolls" />
    <option value="Pesto Artichoke Pizza" />
    <option value="Sal's Stuffed Pizza" />
    <option value="Wing'd Pizza" />
</datalist>
```