# Validating Form Data

# Form Element

- Forms are one of the most common Web page elements used with JavaScript.
- You use JavaScript to make sure that data was entered properly into the form fields and to perform other types of preprocessing before the data is sent to the server.
- You can include as many forms as you like on a Web page, but you cannot nest one form inside another form
- Form Attributes
  - **accept-charset** specifies a comma-separated list of possible character sets that the form supports.
  - **action** is a required attribute that specifies a URL to which form data is submitted. If this attribute is excluded, the data is sent to the URL that contains the form. Typically, you would specify an e-mail address or the URL of a program on a server.
  - **enctype** specifies the MIME type of the data being submitted. The default value is application/x-www-form-urlencoded.
  - **method** determines how form data is submitted. The two options for this attribute are “get” and “post”.

# Attributes of the <form> element

- The enctype attribute is important because a server-side scripting program can use its value to determine how to process the form data.
  - **application/x-www-form-urlencoded** specifies that form data should be encoded as one long string. The format is the same as URL query string.
  - **multipart/form-data** encodes each field as a separate section. Used for submitting forms that contain files, non-ASCII data, and binary data.
  - **text/plain** is used to upload a document to a Web server.
- You can set up a form to send data to an e-mail address. Example,

```
action = "mailto:reservations@npu.edu,
    admissions@npu.edu?
    cc = group_tours@npu.edu,
    subject = group reservations"
```

- When using the **mailto** protocol, be sure to use the enctype of “text/plain”, which can make sure the data arrives at the recipient's e-mail address in a readable format.

# Form Controls

- There are four primary elements used within the <form> element to create form controls: <input>, <button>, <select>, and <textarea>.
- The <input> and <button> elements are used to create input fields with which users interact.
- The <select> element displays choices in a drop-down menu or in a scrolling list known as a selection list.
- The <textarea> element is used to create a text field in which users can enter multiple lines of information.
- The <input>, <textarea>, and <select> elements can include name and value attributes. The name attribute defines a name for an element, and the value attribute defines a default value. When you submit a form to a Web server, the form data is submitted in name = value pairs,
- <input type="text" name="company_info" value="Apple Computer" />
  - name = value pair will be “company_info = Apple Computer”

# JavaScript with Forms

- JavaScript is often used with forms to validate or process form data before the data is submitted to a server-side script.
- To use JavaScript to access form controls and verify form information, you use the Form object, which represents a form on a Web page.
- For instance, the Document object includes a forms[] array that contains all the forms on a Web page.
  - The first form in a document is referred to as document.forms[0], the second form is referred to as document.forms[1], and so on.
- Just as the Document object has a forms[] array, the Form object has an elements[] array. You can use it to reference each element on a form.
- For example, if you want to refer to the first element in the first form on a Web page, use the statement document.forms[0].elements[0];. The third element in the second form is referenced using the state- ment document.forms[1].elements[2];.
- if you want to submit a form to a server-side script, you must include a name attribute for each form element.
- Naming an element also gives you an alternative to referencing the element by its position in the elements[] array,
- For example, if you have an element named quantity in the first form on a Web page, you can refer to it using the statement document.forms[0].quantity;

# Input field object methods

| Method | Description | Form contols |
|---|---|---|
| blur() | Removes focus from a form control | Check boxes, radio buttons, reset buttons, submit buttons, text boxes, text areas, password boxes, file boxes |
| click() | Activates a form control's click event | Check boxes, radio buttons, reset buttons, submit buttons |
| focus() | Changes focus to a form control | Check boxes, radio buttons, reset buttons, submit buttons, text boxes, password boxes, file boxes |
| select() | Selects the text in a form control | Text boxes, password boxes, file boxes |

# Text Boxes

- The following code shows an example of some text boxes that include name, value, and size attributes.
- **value** attribute specifies text to be used as the default value at the moment a form first loads.
- **size** sets or returns a field’s width (in characters)
- **maxLength** sets or returns the maximum number of characters that can be entered into a field

```
<p>Name<br />
<input type="text" name="name" value="Office of the Mayor"
size="50" /></p>
<p>Address<br />
<input type="text" name="address" value="City Hall" size="50" /></p>
<p>City, State, Zip<br />
<input type="text" name="city" value="New York" size="30" />
<input type="text" name="state" value="NY" size="2" maxlength="2" />
<input type="text" name="zip" value="38116" size="10"
maxlength="10" /></p>
```

# isNaN()

- For any fields that require numeric values, for instance, you can use JavaScript's built-in isNaN() function to determine whether the value entered by the user is a number.
- Example,

```
function checkForNumber(fieldValue) {
    • var numberCheck = isNaN(fieldValue);
  if (numberCheck == true) {
     window.alert("You must enter a numeric value!");
     return false;
  }
        else
            -return true;
}
```

In HTML,

```
<input type="text" name="zip" value="38116" size="10" maxlength="10"
onblur="return checkForNumber(this.value);" />
```

# Password Boxes

- An <input> element with a type of “password” (<input type = "password" />) creates a password box that is used for entering passwords or other types of sensitive data. Each character that a user types in a password box appears as an asterisk or bullet, depending on the operating system and Web browser.
- Example, The following code creates a password box with a maximum length of eight characters

```
<p>Please enter a password of <br />
8 characters or less:<br />
<input type="password" name="password" maxlength="8" /></p>
```

- You normally will ask the user to enter the password again in a confirmation field to verify it

```
<p>Confirm password<br />
<input type="password" name="password_confirm"
maxlength="8" onblur="confirmPassword();" /></p>
```

- Use the JavaScript code to check it.

```
function confirmPassword() {
   if (document.forms[0].password_confirm.value !=
document.forms[0].password.value) {
      window.alert("You did not enter the same password!");
      document.forms[0].password.focus();
   }
}
```

# Push Buttons

- An <input> element with a type of “button” (<input type = "button" />) creates a push button that is similar to the OK and Cancel buttons you see in dialog boxes.
- The primary purpose of push buttons is to execute JavaScript code that performs some type of function, such as a calculation.
- Example

```
<p><input type="button" name="push_button"
value="Click Here"
onclick="window.alert('You clicked a push
button.');" />
</p>
```

# Radio Buttons

- An <input> element with a type of "radio" (<input type ="radio" />) is used to create a group of radio buttons, or option buttons, from which the user can select only one value.
- To create a group of radio buttons, all radio buttons in the group must have the same **name** attribute.
- Each radio button requires a **value** attribute that identifies the unique value associated with that button.
- Only the one selected radio button in a group creates a name = value pair when a form is submitted to a Web server.
- You can also include the **checked** attribute in a radio <input> element to set an initial value for a group of radio buttons.
- Example

```
<p>What is your current marital status?<br />

<input type="radio" name="marital_status" value="single" />Single<br />

<input type="radio" name="marital_status" value="married" checked="checked"
/>Married<br />

<input type="radio" name="marital_status" value="divorced" />Divorced<br />

<input type="radio" name="marital_status" value="separated" />Separated<br />

<input type="radio" name="marital_status" value="widowed" />Widowed</p>
```

# Array of elements

- When multiple form elements share the same name, JavaScript creates an array out of the elements using the shared name. Radio buttons, for instance, share the same name so that a single name = value pair can be submitted to a server-side script.
- You can use the following statement to access the value of the second radio button in the group:
- document.forms[0].orderStatus[1].value;
- you can use a statement similar to the following to determine if the first radio button in the group is selected:
- document.forms[0].orderStatus[0].checked;
- Consider the  example you need to check whether the customer wants to be automatically billed on a monthly or yearly basis, rather than pay for a specific number of issues. When the user selects a button, any selected radio button in the payment per number of issues group should be deselected.

```
function billAutomatically() {
   for (var i = 0; i < document.forms[0].delivery.length; ++i){
      if (document.forms[0].delivery[i].checked == true) {
          document.forms[0].delivery[i].checked = false;
          break;
      }
   }
}
```

# Checked Boxes

- An <input> element with a type of “checkbox” (<input type = "checkbox" />) creates a box that can be set to Yes (checked) or No (unchecked).
- You use check boxes when you want users to select whether or not to include a certain item or to allow users to select multiple values from a list of items.
- If a check box is selected (checked) when a form is submitted, then the check box name = value pair is included in the form data. Otherwise it is not included.

```
<h3>Which committees would you like to serve on? </h3>
<p>
    <input type="checkbox" name="committees"
       value="program_dev" />Program Development<br />
    <input type="checkbox" name="committees"
       value = "fundraising" checked />Fundraising<br />
    <input type="checkbox" name="committees"
       value="pub_relations" />Public Relations <br />
    <input type="checkbox" name="committees"
       value="education" />Education
</p>
```

# Selection Lists

- The <select> element creates a selection list that presents users with fixed lists of options created with <option> elements.
- The attributes of the <select> element:
  - **disabled** - disables the selection list
  - **multiple** – allows to select more than one one option.
  - **name** – designates a name for the selection list.
  - **size** – determines how many lines of the selection list appear.
- The attributes of the <option> element:
  - **disabled** - Disables the option
  - **label** - Designates alternate text to display in the selection list for an individual option
  - **selected** - Determines if an option is initially selected in the selection list when the form first loads
  - **value** - Specifies the value submitted to a Web server

# The Select Object and Option Object

- The Select object represents a selection list in a form. The Select object includes an options[] array containing an Option object for each <option> element in the selection list.
- The properties of the Select object are:
  - **disabled** - sets or gets a Boolean value that determines whethere the control is disabled
  - **form** - returns a reference to the form that contains the control
  - **length** - returns the number of elements in the options[] array
  - **multiple** - sets or returns a Boolean value that determines whether multiple selection is allowed.
  - **name** - sets or returns the value assigned to the element’s name attribute
  - **Options**[] - returns an array of the options in a selection list
  - **selectedIndex** - returns a selected option's index if any one; otherwise; returns -1
  - **size** - sets or returns the number of options to display.
- The methods of the Select object
  - **add(element, before)** - adds a new option to a selection list
  - **blur()** - removes focus from a form control
  - **focus()** - changes focus to a form control
  - **remove(index)** - removes an option from a selection list

# The Option Object

- The properties of the Option object:
  - **disabled** - sets or returns a Boolean value that determines whether a control is disabled
  - **form** - returns a reference to the form that contains the Control
  - **index** - returns a number representing the element number within the options[] array
  - **selected** - sets or returns a Boolean value that determines whether an option is selected
  - **text** - sets or returns the text displayed for the option in the selection list
  - **value** - sets or returns the text that is assigned to the <option> element’s value attribute; this is the value that is submitted to the server
  -

# The Option Object

- The properties of the Option object:
  - **DefaultSelected –** set or returns the default value of the selected attribute. So if you reset a form, the value would remain selected.
  - **disabled** - sets or returns a Boolean value that determines whether a control is disabled
  - **form** - returns a reference to the form that contains the Control
  - **index** - returns a number representing the element number within the options[] array
  - **selected** - sets or returns a Boolean value that determines whether an option is selected
  - **text** - sets or returns the text displayed for the option in the selection list
  - **value** - sets or returns the text that is assigned to the <option> element’s value attribute; this is the value that is submitted to the server
  -

# Adding or Removing Options to a Selection List

- The add() method of the Select object to add a new options to a selection list. But this method is not consistently implemented in different web browsers.
- To add a new option to a selection list after a web page is loaded, you need to create a new option with one of Option() constructor.
  - var variable_name = new Option(text, value,defaultSelected, selected);
- To remove an option from a selection list, you pass the option’s index number in the options[] array to the remove() method of the Select object. Examples:
  - document.forms[0].brand.remove(2); // remove the third option
  - document.forms[0].brand.length = 0; // remove all opitons
- To change an option in a selection list, you can just simply assign a new value to option's value or text properties.
  - document.forms[0].brand.options[2].value = “New Value”;
  - document.forms[0].brand.options[2].text = “New text”;

# Example of Select Object and Option Object

```
  <!DOCTYPE html>
• <html>
•     <head>
•         <title>Selection List Example</title>
•         <meta charset="UTF-8">
•         <meta name="viewport" content="width=device-width">
•         <script>
•             function checkData() {
•                 var message = "The number of brands: " + document.forms[0].brand.length + "\r";
•                 var selectedIndex = document.forms[0].brand.selectedIndex;
•                 if (selectedIndex == -1)
•                     message += "No brand s selected.\r";
•                 else
• message += document.forms[0].brand.options[selectedIndex].text + " is selected.\r";
•
•
•                 document.forms[0].info.value = message;
•             }
• function addBrand() {
• document.forms[0].brand.selectedIndex = -1;
• var newBrand = new Option("Fivestars", "Fivestars", false, true);
• document.forms[0].brand.add(newBrand, 2);
• }
• function removeBrand() {
• document.forms[0].brand.selectedIndex = 0;
• document.forms[0].brand.remove(2);
• }
•
•         </script>
•     </head>
•     <body>
•         <h1>Great Computer Store</h1>
•         <h2>Laptop</h2>
•         <form action = "nowhere.html" method = "get">
•             <table border = "0">
•                 <tr>
•                     <td style = "background:white;border:0"><strong>Brand</strong></td>
•                     <td style = "background:white;border:0"><strong>Price Range</strong></td></tr>
•                 <tr>
•                     <td>
•                         <select name = "brand" multiple = "multiple" size = "6">
•                             <option value = "all" selected = "selected">All</option>
•                             <option value = "Apple">Apple</option>
•                             <option value = "Lenovo">Lenovo</option>
•                             <option value = "HP">HP</option>
•                             <option value = "samsung">Samsung</option>
•                             <option value = "Dell">Dell</option>
•                             <option value = "sony">Sony</option>
•                         </select>
•                     </td>
•                     <td>
•                         <select name = "price" size = "6">
•                             <option value = "299">Under $300</option>
•                             <option value = "599">$300 to $599</option>
•                             <option value = "999">$600 to $999</option>
•                             <option value = "1999">$1,000 to $1,999</option>
•                             <option value = "2999" selected = "selected">$2,000 to $2,999</option>
•                             <option value = "3000_plus">Over $3,000</option>
•                         </select>
•                     </td>
•                 </tr>
•             </table>
•             <input type="button" name="submit" value="Submit" onclick="checkData()"/><br>
• <input type="button" name="add" value="Add New Brand" onclick="addBrand()"/><br>
• <input type="button" name="remove" value="Remove Brand" onclick="removeBrand()"/><br>
•             <hr>
•             <textarea name="info" rows="10" cols="80"></textarea>
•         </form>
•     </body>
• </html>
•
```