Forms

Chapters 5
Objectives

• Study form elements and objects
• Use JavaScript to manipulate and validate form elements
• Learn how to submit and reset forms
• Learn how to validate submitted form data
Overview of Forms

• Many Web sites use **forms**
  – Collect information from users and transmit to a server for processing

• Forms frequently found on Web pages gather search criteria from a user
  – Data collected is transmitted to a server-side scripting language program on a Web server

• Popular server-side scripting languages
  – PHP, Common Gateway Interface (CGI), Active Server Pages (ASP), and Java Server Pages (JSP)
Overview of Forms (continued)

Facebook helps you connect and share with the people in your life.

Sign Up
It's free, and always will be.

First Name:
Last Name:
Your email address:
Reenter email address:
New Password:
I am: Select Gender:
Birthday: Day: Month: Year:

Why do I need to provide this?

Sign Up
Overview of Forms (continued)

My Opodo for Mr Enrico Gerding

My trips

My profile

Saving your travel preferences helps you save time during the planning and booking process.

Personal details and travel options

Required fields are marked with an asterisk, but the more information you save here in your profile, the less information you have to type each time you book.

- Title*: Mr
- First name*
- Surname*
- Email*
- Date of birth*: 1 1 1974
- Main contact no (optional)
- Traveller type: Adult (optional)
- Special meals: None (optional)

I confirm that I have obtained the consent of the traveller named above for this information to be used in accordance with Opodo’s privacy and security policies.
The `<form>` Element

- `<form>` element
  - Designates a form within a Web page and contains all the text and elements that make up a form
Form Elements

- Primary elements used within the `<form>` element
  - `<input>`, `<button>`, `<select>`, and `<textarea>`
- `<input>` and `<button>` elements
  - Create input fields with which users interact
- `<select>` element
  - Displays choices in drop-down menu or scrolling list
- `<textarea>` element
  - Create a text field in which users can enter multiple lines of information
Form Elements

• `<input>`, `<textarea>`, and `<select>` elements can include name and value attributes
  – name attribute defines a name for an element
  – value attribute defines a default value

• These values are used when forms are submitted to server
Examples

<form>
    <input type="text"
    name="firstName" />
    <input type="text"
    name="lastName" />
</form>
Working with Input Fields

• Types of input fields
  – button
  – text
  – radio
  – checkbox
  – file
  – image
  – submit
  – password
  – hidden
  – reset
Working with Input Fields (continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept</td>
<td>Determines the MIME type of a document that is uploaded with a file box</td>
</tr>
<tr>
<td>alt</td>
<td>Provides alternate text for an image submit button</td>
</tr>
<tr>
<td>checked</td>
<td>Determines whether or not a radio button or a check box is selected; a Boolean attribute</td>
</tr>
<tr>
<td>disabled</td>
<td>Disables a control</td>
</tr>
<tr>
<td>maxlength</td>
<td>Accepts an integer value that determines the number of characters that can be entered into a field</td>
</tr>
<tr>
<td>name</td>
<td>Designates a name for the element; part of the name=value pair that is used to submit data to a Web server</td>
</tr>
<tr>
<td>readonly</td>
<td>Prevents users from changing values in a control</td>
</tr>
<tr>
<td>size</td>
<td>Accepts an integer value that determines the width of a text box in characters</td>
</tr>
<tr>
<td>src</td>
<td>Specifies the URL of an image</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of element to be rendered; type is a required attribute; valid values are text, password, radio, check box, reset, button, submit, image, file, and hidden</td>
</tr>
<tr>
<td>value</td>
<td>Sets an initial value in a field or a label for buttons; part of the name=value pair that is used to submit data to a Web server</td>
</tr>
</tbody>
</table>

Table 5-5 Attributes of the `<input>` element
Text Boxes

• Text box
  – An `<input>` element with a type of “text”
  – Accepts a single line of text

• value attribute
  – Specifies text to be used as the default value at the moment a form first loads
Password Boxes

• Password box
  – An `<input>` element with a type of “password”
  – Entering passwords or other types of sensitive data
  – Character typed appears as an asterisk or bullet
Push Buttons

• **Push button**
  
  – An `<input>` element with a type of “button”
  – Similar to OK and Cancel buttons in dialog boxes
  – Primary purpose is to execute JavaScript code

• **Example**

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

• Group of **radio buttons**, or **option buttons**
  – An `<input>` element with a type of “radio”
  – User can select only one value
• All radio buttons in the group must have the same **name** attribute
• Each radio button requires a **value** attribute that identifies its unique value
• **checked** attribute in a radio `<input>` element
  – Sets an initial value for the group
Example

<form> Please enter your sex:
  Male <input type="radio" name="sex" value="male" />
  Female <input type="radio" name="sex" value="female" checked />
</form>
Check Boxes

Figure 5-11
Form with check boxes
Check Boxes

• **Check boxes**
  – An `<input>` element with a type of “checkbox”
  – Can be set to Yes (checked) or No (unchecked)
  – 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

• `checked` attribute
  – Sets the initial value of the check box to Yes

• Group check boxes by giving each check box the same name value
Example

<form> Please uncheck this box if you want to receive even more SPAM from our sponsors
<input type="checkbox" name="spam" checked />
</form>
Accessing Forms and Form Elements

• Several ways to access elements
  – Using DOM: `getElementById`, `getElementsByTagName`, `getElementsByClassName`, etc.
  – Using JavaScript Objects
Using JavaScript with Forms

• For each `<form>` tag, a JavaScript `form` object is created
• `document` object includes a `forms[]` array that contains all the forms on a Web page
• `form` elements:
  – Each `form` object has an `elements[]` array
  – `elements[]` array
    • Contains objects representing each control in a form
• `input` element properties
  – `name`, `value`, `defaultValue`, `type`, `form`, ...
Example

<form>
<p>Enter your name: <input type="text" name="myName" /></p>
<input type="button" value="Press Me"
onclick="window.alert('Your name is:' + document.forms[0].elements[0].value)"/>
</form>
Alternatives

• Use `getElementById()`

• Use the keyword `this`:

```html
onclick="window.alert('Your name is:' + this.form.elements[0].value)" />
```

*note: need to access form elements via the `form` object of the html element*
Alternative (2)

• For each HTML elements with a `name` attribute, a JavaScript object property is created with the same name
• If there are multiple HTML elements with the same `name` attribute, the object property will be an array
• `name` attribute should only be used for form elements, and has been deprecated for all other html tag elements!
Example

<form>
  <p>Enter your name: <input type="text" name="myName" /> </p>
  <input type="button" value="Press Me" onclick="window.alert('Your name is: '+this.form.myName.value)"/>
</form>
Example

<form>
<p>Enter your first name: <input type="text" name="myName" /></p>
<p>Enter your last name: <input type="text" name="myName" /></p>
<input type="button" value="Press Me" onclick="window.alert('Your name is:' + this.form.myName[0].value + ' ' + this.form.myName[1].value)" />
</form>
Validating Radio Buttons and Check Boxes

• Radio buttons
  – If multiple buttons with same name, element consists of an array
  – Use the `checked` property to determine which element in a group is selected

• Check boxes
  – Use the `checked` property to determine whether an individual check box has been selected
Example: Quiz

<h1>Strange Quiz</h1>

<form>
  <p><strong>Question 1: What is the answer?</strong></p>
  <p><input type="radio" name="question1" value="a"/>Answer is A</p>
  <p><input type="radio" name="question1" value="b"/>Answer is B</p>
  <p><input type="radio" name="question1" value="c"/>Answer is C</p>
  <input type="button" value="Submit" onclick="scoreQuestion(this.form.elements)"/>
</form>
Example (cont.)

```javascript
function scoreQuestion(form_elements)
{
   for (var i=0; i < form_elements.length; i++) {
      if (form_elements[i].checked)
         answer = form_elements[i].value;
   }
   if (answer == "a")
      window.alert("wrong");
   if (answer == "b")
      window.alert("right");
   if (answer == "c")
      window.alert("wrong");
}
```
Validating Submitted Data

• Validate form before sending to server
  – Always need to validate forms
  – Can be done either with JavaScript or on server side

• `onsubmit` event handler
  – Executes when a form is submitted to a server-side script

• Must return a value of `true` or `false`
  – Depending on whether the form should be submitted (true) or reset (false)
Validating Text and Password Boxes

• Use an `if` statement in the `onsubmit` event handler
  – Check whether field’s value property contains a value
Example

<form onsubmit="return validateForm()">
  Name: <input type="text" name="name" />
  Password: <input type="password" name="password" />
</form>

... 

function validateForm() {
  if (document.forms[0].elements[0].value == "" ||
      document.forms[0].elements[1].value == "") {
    window.alert("Missing name or password");
    return false;
  }
  return true;
}
Radio Button and Check Boxes Example

<form onsubmit="return validateForm()">
  Sex: Male <input type="radio" name="sex" value="male" />
  Female <input type="radio" name="sex" value="female" />
  <input type="submit" value="Submit Form"/>
</form>

function validateForm() {
  if (document.forms[0].elements[0].checked==false &&
      document.forms[0].elements[1].checked==false) {
    window.alert("You have not made any selection");
    return false;
  }
  return true;
}
Submitting a Form

• Main purpose of form is ability to submit information to server or other webpage for further processing

• Use `<form>` attributes:
  
  `action` - specifies a **URL** to which form data is submitted (can also be email address)
  
  `enctype` - MIME type of data being submitted
  
  `method` - `get` or `post`
Submitting a Form

• Submit button
  – An `<input type="submit">` element
  – Transmits a form’s data to a Web server

• Call `submit()` function

• activates `action` attribute of the `<form>` element
  – Determines to what URL the form is submitted
  – Only those form elements with a `name attribute`
  – Format: value=name
Examples

• Send form content to FormProcessor.html
  <form action="FormProcessor.html" enctype="application/x-www.form-urlencoded" method="get">

• Send form content to email address:
  <form enctype="text/plain" action="mailto:eg@ecs.soton.ac.uk?subject=some text" method="post">

37
<form enctype="text/plain"
  action="mailto:eg@ecs.soton.ac.uk"
  method="post">
    <p>First Name: <input type="text"
        name="firstname" /></p>
    <p>Last Name:  <input type="text"
        name="lastname" /></p>
    <p><input type="submit" /></p>
</form>
method - get

- This method appends the form-data to the URL in name/value pairs
  - http://.../FormProcessor.html?firstname=Enrico&lastname=Gerding

- This method is useful for form submissions where a user want to bookmark the result

- There is a limit to how much data you can place in a URL (varies between browsers), therefore, you cannot be sure that all of the form-data will be correctly transferred

- Never use the "get" method to pass sensitive information! (password or other sensitive information will be visible in the browser's address bar)
method - post

• This method sends the form-data as an HTTP post transaction
• Typically used to send data to the server for further processing
  – e.g. using PHP or ASP.NET
• Form submissions with the "post" method cannot be bookmarked
• The "post" method is more robust and secure than "get", and "post" does not have size limitations
Summary

• Forms collect information from users and transmit that information to a server for processing

• Important to validate forms
What's next?

• Forms contain information on a specific web site
• What if the form consists of multiple pages?
• What if we want to remember what a user has filled in earlier?
• How? We need to pass on or store "state" information
State information

• Information about individual visits to a web site

• HTTP was originally designed to be stateless
  – Browsers stored no persistent data about site visits
Why keep state information?

• Customize individual Web pages
• Temporarily store information for a user
• Allow a user to create bookmarks
• Provide shopping carts
• Store user IDs and passwords
• Use counters
Approaches

• Hidden field elements
• Query passing
• Cookies
Optional slides
Creating Selection Lists

• The `<select>` element creates a selection list
  – Presents users with fixed lists of options
• Options displayed in a selection list are created with `<option>` elements
• `<select>` element must appear within a block-level element such as the `<p>` element
• A selection list can also include a scroll bar
Creating Selection Lists (continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disabled</td>
<td>Disables the selection list</td>
</tr>
<tr>
<td>multiple</td>
<td>Specifies whether a user can select more than one option from the list; a Boolean attribute</td>
</tr>
<tr>
<td>name</td>
<td>Designates a name for the selection list</td>
</tr>
<tr>
<td>size</td>
<td>Determines how many lines of the selection list appear</td>
</tr>
</tbody>
</table>

Table 5-8  Attributes of the `<select>` element
Menu Options

• `<option>` element
  – Specifies the options that appear in a selection list
• Each selection list must contain at least one `<option>` element
• Example: The Gosselin Gazette Web page
  – Add a selection list to Subscription.html
    • Subscriber uses to select any magazines to which they are currently subscribed
Menu Options (continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disabled</td>
<td>Disables the option</td>
</tr>
<tr>
<td>label</td>
<td>Designates alternate text to display in the selection list for an individual option</td>
</tr>
<tr>
<td>selected</td>
<td>Determines if an option is initially selected in the selection list when the form first loads; a Boolean attribute</td>
</tr>
<tr>
<td>value</td>
<td>Specifies the value submitted to a Web server</td>
</tr>
</tbody>
</table>

*Table 5-9*  Attributes of the `<option>` element
The Select and Option Objects

• **Select object**
  – Represents a selection list in a form
  – Includes an `options[]` array containing an `Option object` for each `<option>` element in the selection list

• **Option object**
  – Represents an option in a selection list
The **Select and Option Objects** (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disabled</td>
<td>Sets or returns a Boolean value that determines whether a control is disabled</td>
</tr>
<tr>
<td>form</td>
<td>Returns a reference to the form that contains the control</td>
</tr>
<tr>
<td>length</td>
<td>Returns the number of elements in the <code>options[]</code> array</td>
</tr>
<tr>
<td>multiple</td>
<td>Sets or returns a Boolean value that determines whether multiple options can be selected in a selection list</td>
</tr>
<tr>
<td>name</td>
<td>Sets or returns the value assigned to the element’s name attribute</td>
</tr>
<tr>
<td>options[]</td>
<td>Returns an array of the options in a selection list</td>
</tr>
<tr>
<td>selectedIndex</td>
<td>Returns a number representing the element number in the <code>options[]</code> array of the first option selected in a selection list; returns -1 if No option is selected</td>
</tr>
<tr>
<td>size</td>
<td>Sets or returns the number of options to display</td>
</tr>
<tr>
<td>tabIndex</td>
<td>Sets or returns a control’s position in the tab order</td>
</tr>
<tr>
<td>type</td>
<td>Returns the type of selection list; returns “select-one” if the <code>&lt;select&gt;</code> element does not include the multiple attribute, or it returns “select-multiple” if the <code>&lt;select&gt;</code> element does includes the multiple attribute</td>
</tr>
</tbody>
</table>

**Table 5-10** Properties of the Select object
The **Select** and **Option** Objects (continued)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>add(element, before)</code></td>
<td>Adds a new option to a selection list</td>
</tr>
<tr>
<td><code>blur()</code></td>
<td>Removes focus from a form control</td>
</tr>
<tr>
<td><code>focus()</code></td>
<td>Changes focus to a form control</td>
</tr>
<tr>
<td><code>remove(index)</code></td>
<td>Removes an option from a selection list</td>
</tr>
</tbody>
</table>

**Table 5-11**  Methods of the `Select` object

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>defaultSelected</code></td>
<td>Returns a Boolean value that determines whether the <code>&lt;option&gt;</code> element representing the currently selected item includes the <code>selected</code> attribute</td>
</tr>
<tr>
<td><code>disabled</code></td>
<td>Sets or returns a Boolean value that determines whether a control is disabled</td>
</tr>
<tr>
<td><code>form</code></td>
<td>Returns a reference to the form that contains the control</td>
</tr>
<tr>
<td><code>index</code></td>
<td>Returns a number representing the element number within the <code>options[]</code> array</td>
</tr>
<tr>
<td><code>label</code></td>
<td>Sets or returns alternate text to display for the option in the selection list</td>
</tr>
<tr>
<td><code>selected</code></td>
<td>Sets or returns a Boolean value that determines whether an option is selected</td>
</tr>
<tr>
<td><code>text</code></td>
<td>Sets or returns the text displayed for the option in the selection list</td>
</tr>
<tr>
<td><code>value</code></td>
<td>Sets or returns the text that is assigned to the <code>&lt;option&gt;</code> element’s value attribute; this is the value that is submitted to the server</td>
</tr>
</tbody>
</table>

**Table 5-12**  Properties of the `Option` object
Adding Options to a Selection List

• ECMAScript recommendations suggest using the `add()` method of the `Select` object
  – To add new options to a selection list
  – Method is not consistently implemented

• Create a new option with `Option()` constructor
  – Then, assign the object to an empty element in an options[] array

• Example
  – Add a selection list to Subscription.html
Removing Options from a Selection List

• Pass option’s index number in `options[]` array to the `remove()` method of the `Select` object
  – Remaining elements are reordered

• Remove all the options from an options array
  – Set `length of options[]` array to zero

• Example
  – Add code to `Subscription.html` that deletes magazine names from the selection list
Changing Options in a Selection List

• Assign new values to the option’s `value` and `text` properties

• Example: The Gosselin Gazette Web Page
  – Add code to Subscription.html that modifies magazine names in the selection list
Submitting and Resetting Forms

• Learn how to submit forms to a server-side script
  – And how to reset form fields to their default values

• Use JavaScript to:
  – Validate submitted data
  – Confirm whether users really want to reset form fields
Submit Buttons

• **Submit button**
  – An `<input>` element with a type of “submit”
  – Transmits a form’s data to a Web server

• **action attribute of the `<form>` element**
  – Determines to what URL the form is submitted

• **Submit buttons do not have values that are submitted to a Web server**

• **Image submit button**
  – An `<input>` element with a type of “image”
  – Displays a graphical image and transmits a form’s data to a Web server
Submit Buttons (continued)

• **Image submit button (continued)**
  – Include the `src` attribute to specify the image to display on the button

• **Example: The Gosselin Gazette Web page**
  – Add a submit button to Subscription.html
Submit Buttons (continued)

**Figure 5-16**
Subscription form after adding an image submit button
Reset Buttons

• **Reset button**
  – An `<input>` element with a type of “reset”
  – Clears all form entries and resets each form element to the initial value specified by its `value` attribute

• Text you assign to the reset button’s `value` attribute appears as the button label

• Example: The Gosselin Gazette Web page
  – Add a reset button to Subscription.html