# SWEN1005

MOBILE WEB PROGRAMMING

# Session Seven

HTML FORMS AND JAVASCRIPT

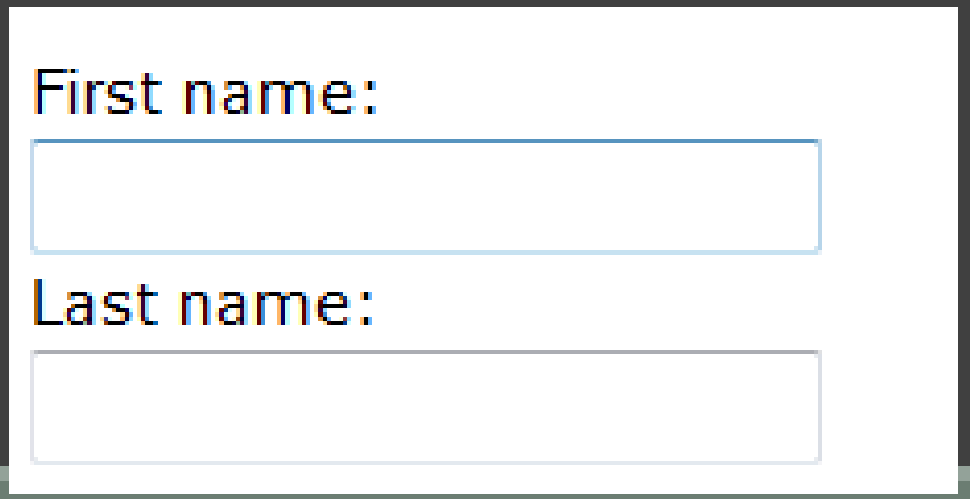
# What are forms?

- <form> is just another kind of HTML tag
- Defines a form usually used to collect user input
- Forms are used to create (rather primitive) GUIs on Web pages
  - Usually the purpose is to ask the user for information
  - The information is then sent back to the server

# What are forms?

- A form is an area that can contain form elements
- The syntax is: <form parameters> ...form elements... </form>
- Form elements include: buttons, checkboxes, text fields, radio buttons, drop-down menus, etc
  - Other kinds of tags can be mixed in with the form elements

```
<form>
  First name:<br>
  <input type="text" name="firstname"><br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```



# What are forms?

- A form is an area that can contain form elements
- A form usually contains a **submit** button to send the information from the form elements to the server
- The form's **parameters** tell JavaScript how to send the information to the server (there are two different ways it could be sent)
- Forms can be used for other things, such as a GUI for simple programs which is how we used it in the last lab and how we will use it for our apps.

# Forms and JavaScript

- You can use JavaScript to create special effects.
- You can use JavaScript to make your HTML pages "smarter" by exploiting its decision-making capabilities.
- We can also use JavaScript to enhance HTML forms. This last application is of particular importance. Of all the hats JavaScript can wear, its form processing features are among the most sought and used.

# Forms and JavaScript

- CGI (which stands for Common Gateway Interface), is a mechanism for safely transporting data from a client (a browser) to a server.
- It is typically used to transfer data from an HTML form to the server.

# Forms and JavaScript

- With JavaScript you can process simple forms without invoking the server
- When submitting the form to a CGI program is necessary, you can have JavaScript take care of all the preliminary requirements, such as validating input

# Forms and JavaScript

- We will look at the JavaScript-form connection
- Using JavaScript's form object
- Reading and setting form content
- Triggering JavaScript events by manipulating form controls
- Using JavaScript to verify form input

# Forms and JavaScript

- The JavaScript language can be used to make pages that "do something"
  - Usually you just use snippets of JavaScript here and there throughout your Web page
  - JavaScript code snippets can be attached to various form elements
  - Check that a zipcode field contains a 5-digit integer before you send that information to the server
- This is sometimes called "active scripting"
- Forms can be used without JavaScript, and JavaScript can be used without forms, but they work well together

# Forms and JavaScript

- There are few differences between a straight HTML form and a JavaScript-enhanced form.
- The main one being that a JavaScript form relies on one or more event handlers, such as onclick or onsubmit.
- These invoke a JavaScript action when the user does something in the form, like clicking a button.

# Forms and JavaScript

- Typical form control objects include:
  - Text box for entering a line of text
  - Push button for selecting an action
  - Radio buttons for making one selection among a group of options
  - Check boxes for selecting or deselecting a single, independent option

| Type | Description |
| --- | --- |
| <input type="text"> | Defines a one-line text input field |
| <input type="radio"> | Defines a radio button (for selecting one of many choices) |
| <input type="submit"> | Defines a submit button (for submitting the form) |

# Forms and JavaScript

- For use with JavaScript, you should always remember to provide a name for the form itself, and each control you use
- The names allow you to reference the object in your JavaScript-enhanced page.

# The <form> tag

- The <form arguments> ... </form> tag encloses form elements (and sometimes other elements as well)
- The arguments tell what to do with the user input
  - **action="url"** (required) where to send the data when the Submit button is clicked
  - **method="get"** (default) Form data is sent as a URL with ?form_data info appended at the end, aka query string
  - Can be used only if data is all ASCII and not more than 100 characters

# The <form> tag

- **method="post“** Form data is sent in the body of the URL request
- Cannot be bookmarked by most browsers
- **target="target“ (_blank or _top)**
- Tells where to open the page sent as a result of the request

# Forms and JavaScript

- *FORM NAME="myform"* defines and names the form. Elsewhere in the JavaScript you can reference this form by the name *myform*.
- The name you give your form is up to you, but it should comply with JavaScript's standard variable/function naming rules

```
1  <FORM NAME="myform" ACTION="" METHOD="GET">
2  Enter something in the box: <BR>
3  <INPUT TYPE="text" NAME="inputbox" VALUE=""><P>
4  <INPUT TYPE="button" NAME="button" Value="Click" onClick="testResults(this.form)">
5  </FORM>
```

# Forms and JavaScript

- *ACTION=“ "* defines how you want the browser to handle the form when it is submitted to a CGI program running on the server. As this example is not designed to submit anything, the URL for the CGI program is omitted.
- *METHOD="GET"* defines the method data is passed to the server when the form is submitted. In this case the attribute is insignificant as the example form does not submit anything.

```
1  <FORM NAME="myform" ACTION="" METHOD="GET">
2  Enter something in the box: <BR>
3  <INPUT TYPE="text" NAME="inputbox" VALUE=""><P>
4  <INPUT TYPE="button" NAME="button" Value="Click" onClick="testResults(this.form)">
5  </FORM>
```

# Forms and JavaScript

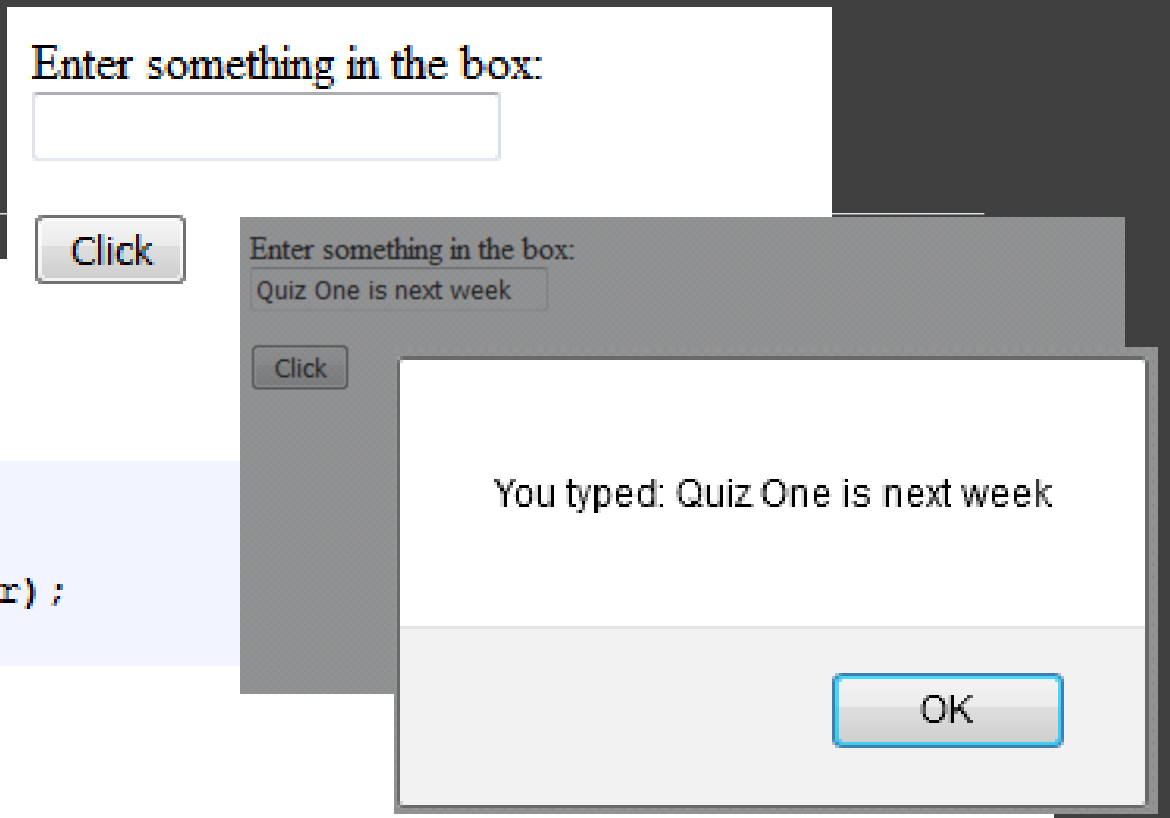
- *INPUT TYPE="text"* defines the text box object. This is standard HTML markup.
- *INPUT TYPE="button"* defines the button object. This is standard HTML markup except for the onclick handler.
- *onclick="testResults(this.form)"* is an event handler -- it handles an event, in this case clicking the button. When the button is clicked, JavaScript executes the expression within the quotes. The expression says to call the testResults function elsewhere on the page, and pass to it the current form object.

# Getting a value from a form object

- So how do we load the page, then type something into the text box.
- Click the button, and have what we typed  show in an alert box.

# Example

```
1  <HTML>
2  <HEAD>
3   <TITLE>Test Input</TITLE>
4  <SCRIPT LANGUAGE="JavaScript">
5  function testResults (form) {
6       var TestVar = form.inputbox.value;
7       window.alert ("You typed: " + TestVar);
8  }
9  </SCRIPT>
10 </HEAD>
11 <BODY>
12 <FORM NAME="myform" ACTION="" METHOD="GET">Enter something in the box: <BR>
13 <INPUT TYPE="text" NAME="inputbox" VALUE=""><P>
14  <INPUT TYPE="button" NAME="button" Value="Click" onClick="testResults(this.form)">
15 </FORM>
16 </BODY>
17 </HTML>
18
```



# Example

- JavaScript calls the testResults function when you click the button in the form.
- The testResults function is passed the form object using the syntax this.form (this keyword references the button control; form is a property of the button control and represents the form object).
- The form object has the name *form* inside the testResult function, but you can any name you like.

# Example

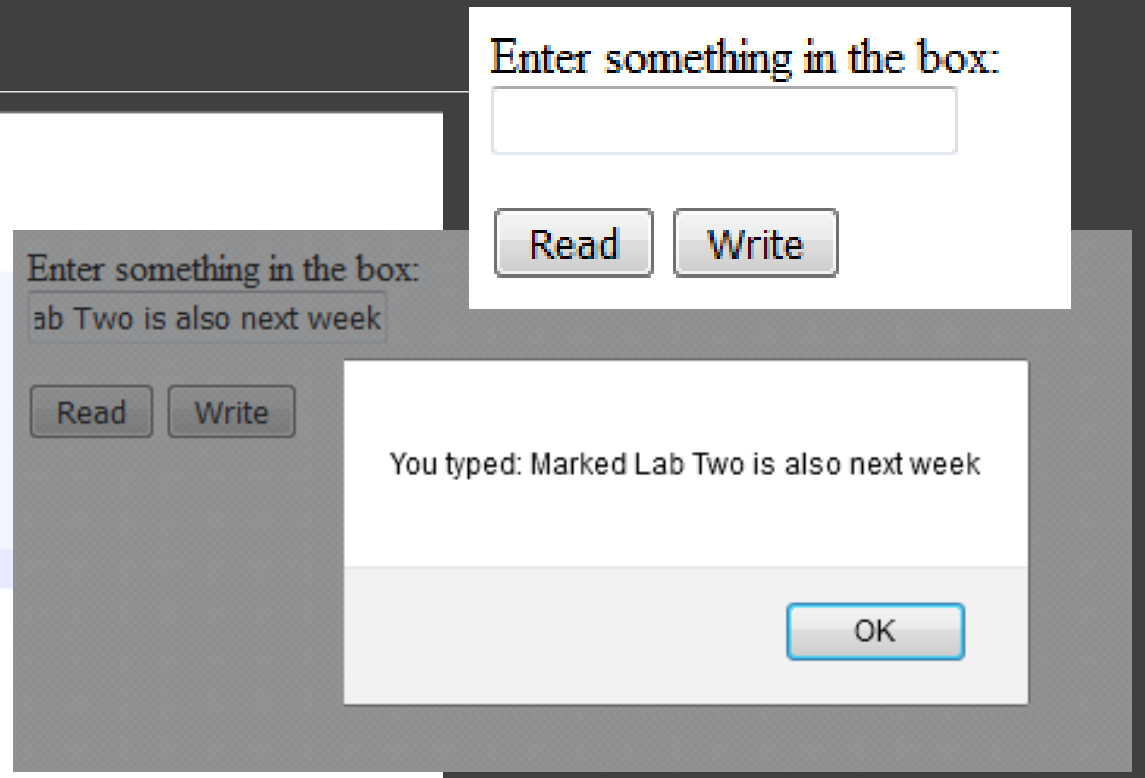
- The testResults function merely copies the contents of the text box to a variable named TestVar.
- Notice how the text box contents was referenced.
- The form object used (called *form*), the object within the form that I wanted (called *inputbox*), and the property of that object wanted (the *value* property).
  - Hence “form.inputbox.value”

# Setting a value in a form object

- The value property of the inputbox, is both readable and writable.
- That is, you can read whatever is typed into the box, and you can write data back into it.
- The process of setting the value in a form object is just the reverse of reading it.

# Example

```
1  <HTML>
2  <HEAD>
3  <TITLE>Test Input </TITLE>
4  <SCRIPT LANGUAGE="JavaScript">
5  function readText (form) {
6      TestVar =form.inputbox.value;
7      window.alert ("You typed: " + TestVar);
8  }
9
10 function writeText (form) {
11     form.inputbox.value = "Have a nice day!"
12 }
13 </SCRIPT>
14 </HEAD>
15 <BODY>
16 <FORM NAME="myform" ACTION="" METHOD="GET">
17 Enter something in the box: <BR>
18 <INPUT TYPE="text" NAME="inputbox" VALUE=""><P>
19 <INPUT TYPE="button" NAME="button1" Value="Read" onClick="readText(this.form)">
20 <INPUT TYPE="button" NAME="button2" Value="Write" onClick="writeText(this.form)">
21 </FORM>
22 </BODY>
23 </HTML
```



# Setting a value in a form object

- The example demonstrated setting a value in a form text box.
- It was similar to the previous example, except this time there are two buttons.
- Click the "Read" button and the script reads what you typed into the text box.
- Click the "Write" button and the script writes a pleasant phrase into the text box.

# Setting a value in a form object

- When you click the "Read" button, JavaScript calls the readText function, which reads and displays the value you entered into the text box.
- When you click the "Write" button, JavaScript calls the writeText function, which writes "Have a nice day!" in the text box.

# Reading other form object values

- JavaScript can read and write values to and from most other objects
- In addition to text boxes, JavaScript can be used with:
  - Hidden text box (TYPE="hidden").
  - Radio button (TYPE="radio")
  - Check box (TYPE="checkbox")
  - Text area (<TEXT AREA>)
  - Lists (<SELECT>)

# Using Hidden Text Boxes

- From a JavaScript standpoint, hidden text boxes behave just like regular text boxes, sharing the same properties and methods.
- From a user standpoint, hidden text boxes "don't exist" because they do not appear in the form.
- Hidden text boxes are the means by which special information can be passed between server and client.
- They can also be used to hold temporary data that you might want to use later.

# Using Radio Buttons

- Radio buttons are used to allow the user to select one, and only one, item from a group of options.
- Radio buttons are always used in multiples; there is no logical sense in having just one radio button on a form, because once you click on it, you can't unclick it.
- If you want a simple click/unclick choice use a check box instead

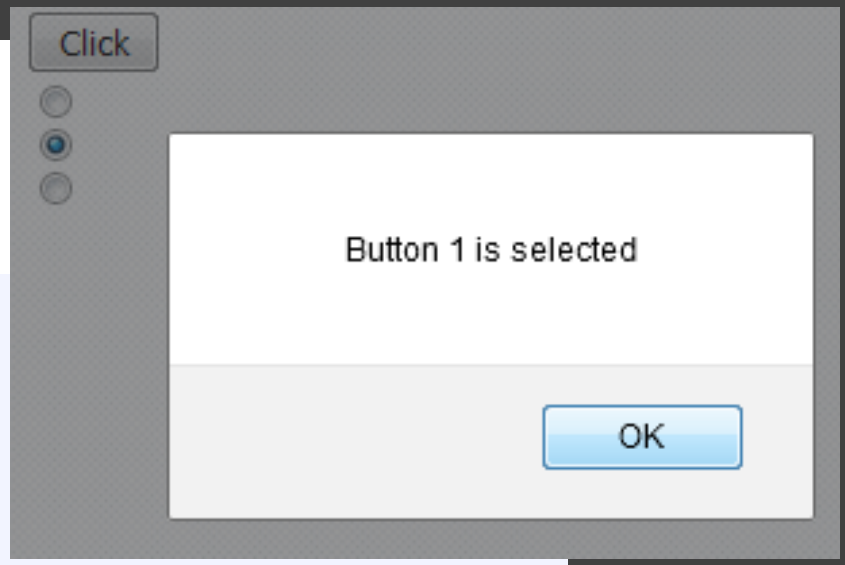
# Using Radio Buttons

- To define radio buttons for JavaScript, provide each object with the same name.
- JavaScript will create an array using the name you've provided; you then reference the buttons using the array indexes.
- The first button in the series is numbered 0, the second is numbered 1, and so forth.
- Note that the VALUE attribute is optional for JavaScript-only forms.

# Example

```
<INPUT TYPE="radio" NAME="rad" VALUE="radio_button1" onClick=0>
<INPUT TYPE="radio" NAME="rad" VALUE="radio_button2" onClick=0>
<INPUT TYPE="radio" NAME="rad" VALUE="radio_button3" onClick=0>
<INPUT TYPE="radio" NAME="rad" VALUE="radio_button4" onClick=0>
```

```
<HTML>
<HEAD>
<TITLE>Radio Button Test</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function testButton (form){
    for (Count = 0; Count < 3; Count++) {
        if (form.rad[Count].checked)
            break;
    }
    alert ("Button " + Count + " is selected");
}
</SCRIPT>
</BODY>
<FORM NAME="testform">
<INPUT TYPE="button" NAME="button" Value="Click"
    onClick="testButton(this.form)"> <BR>
<INPUT TYPE="radio" NAME="rad" Value="rad_button1" onClick=0><BR>
<INPUT TYPE="radio" NAME="rad" Value="rad_button2" onClick=0><BR>
<INPUT TYPE="radio" NAME="rad" Value="rad_button3" onClick=0><BR>
</FORM>
</HTML>
```



# Example

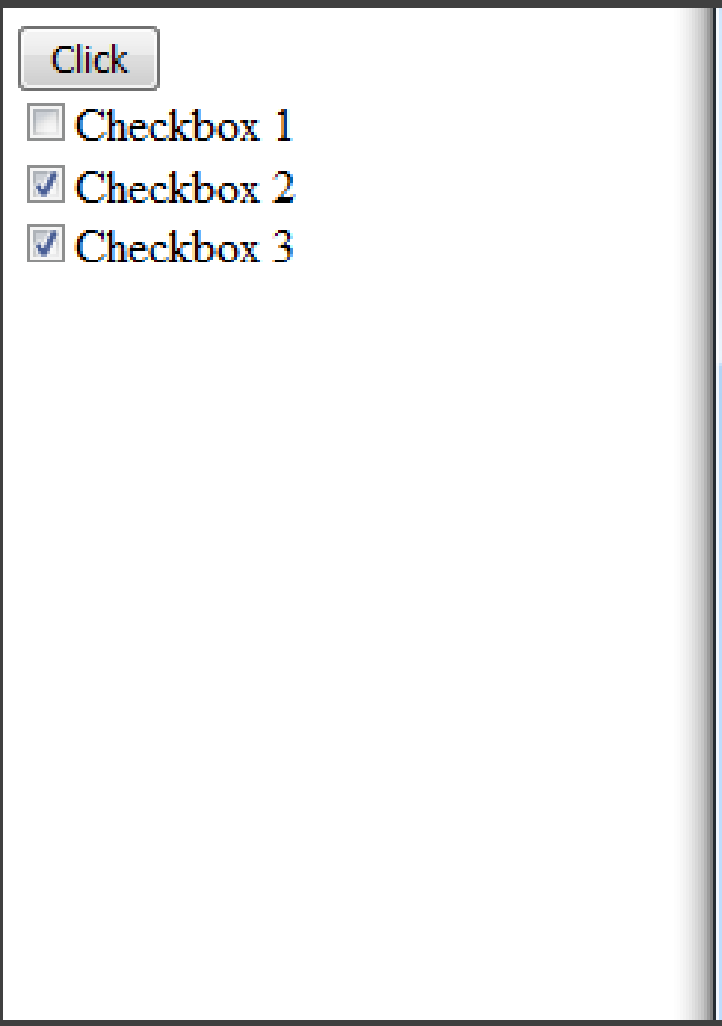
- In the example  we are testing which button is selected
- The for loop in the testButton function cycles through all of the buttons in the "rad" group
- When it finds the button that's selected, it breaks out of the loop and displays the button number (remember: starting from 0).

# More on radio buttons

- Setting a radio button selection with HTML market is simple. If you want the form to initially appear with a given radio button selected, add the CHECKED attribute to the HTML markup for that button:
  - <INPUT TYPE="radio" NAME="rad" Value="rad_button1" CHECKED onClick=0>
- You can also set the button selection programmatically with JavaScript, using the checked property. Specify the index of the radio button array you want to checked.
  - form.rad[0].checked = true; // sets to first button in the rad group

# Using Check Boxes

- Check boxes are stand-alone elements; that is, they don't interact with neighboring elements like radio buttons do
- Using JavaScript you can test if a check box is checked using the checked property
- Likewise, you can set the checked property to add or remove the checkmark from a check box.




HomePage.html | LabFive | form.html

```
1  <HTML>
2  <HEAD>
3  <TITLE>Checkbox Test</TITLE>
4  <SCRIPT LANGUAGE="JavaScript">
5  function testButton (form){
6      window.alert (form.check1.checked);
7  }
8  </SCRIPT>
9  </BODY>
10 <FORM NAME="testform">
11 <INPUT TYPE="button" NAME="button" Value="Click"
12     onClick="testButton(this.form)"><BR>
13 <INPUT TYPE="checkbox" NAME="check1" Value="Check1">Checkbox 1<BR>
14 <INPUT TYPE="checkbox" NAME="check2" Value="Check2">Checkbox 2<BR>
15 <INPUT TYPE="checkbox" NAME="check3" Value="Check3">Checkbox 3<BR>
16 </FORM>
17 </BODY>
18 </HTML>
```

# More on checkboxes

- As with the radio button object, add a CHECKED attribute to the HTML markup for that check box you wish to be initially check when the form is first loaded.
  - <INPUT TYPE="checkbox" NAME="check1" Value="0" CHECKED>Checkbox 1>
- You can also set the button selection programmatically with JavaScript, using the checked property. Specify the name of the checkbox you want to check, as in
  - form.check1.checked = true;

# Using Text Areas

- Text areas are used for multiple-line text entry.
- The default size of the text box is 1 row by 20 characters.
- You can change the size using the COLS and ROWS attributes.
- Example
  - <TEXTAREA NAME="myarea" COLS="40" ROWS="7"> </TEXTAREA>
- You can use JavaScript to read the contents of the text area box. This is done with the value property.

here is some blah blah text
xdkkjfdkfn
sdfdf
dfdsfgsdf
dfsdf
sdfs

Test

HomePage.html | LabFive | form.html

```
1  <HTML>
2  <HEAD>
3  <TITLE>Text Area Test</TITLE>
4  <SCRIPT LANGUAGE="JavaScript">
5  function seeTextArea (form) {
6      alert (form.myarea.value);
7
8  }
9  </SCRIPT>
10 </HEAD>
11 <BODY>
12 <FORM NAME="myform">
13 <INPUT TYPE="button" NAME="button3" Value="Test"
14     onClick="seeTextArea(this.form)">
15 <TEXTAREA NAME="myarea" COLS="40" ROWS="5">
16 </TEXTAREA>
17 </FORM>
18 </BODY>
19 </HTML>
```

# More on textareas

- You can preload text into the text area in either of two ways. One method is to enclose text between the <TEXTAREA> and </TEXTAREA> tags.
- This method is useful if you wish to include hard returns, as these are retained in the text area box. Or, you can set it programmatically with JavaScript using the following syntax:
  - form.textarea.value = "Text goes here"; *form* is the name of the form.
  - *textarea* is the name of the textarea.
  - *"Text goes here"* is the text you want to display

# Using Selection Lists

- List boxes let you pick the item you want out of a multiple-choice box.
- The listbox itself is created with the <SELECT> tag, and the items inside it are created by one or more <OPTION> tags.
- You can have any number of <OPTION> tags in a list. The list is terminated with a </SELECT> tag.

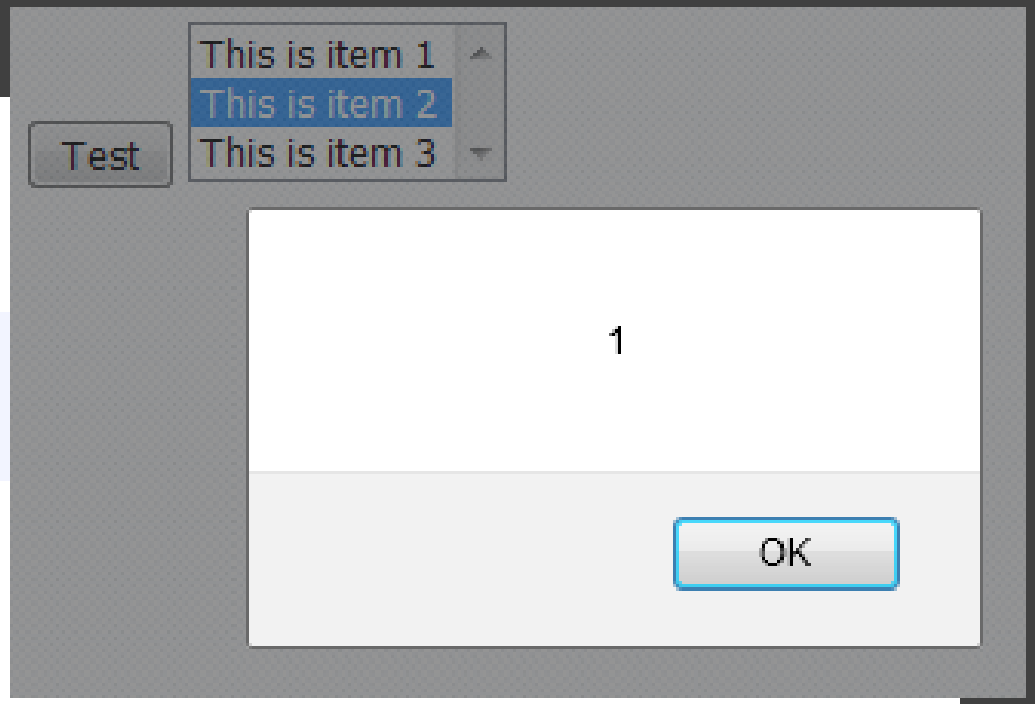
# Using Selection Lists

- The list can appear with many items showing at once, or it can appear in a drop-down box -- normally you see one item at a time, but click to see more.
- The markup for the two styles is identical, except for the optional SIZE attribute.
- Leave off SIZE to make a drop-down box; use SIZE to make a list box of the size you wish.

# Using Selection Lists

- Use the selectedIndex property to test which option item is selected in the list, as shown in the following example.
- The item is returned as an index value, with 0 being the first option, 1 being the second, and so on
- If no item is selected the value is -1

```
<HTML>
<HEAD>
<TITLE>List Box Test</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function testSelect(form) {
    alert (form.list.selectedIndex);
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="myform" ACTION="" METHOD="GET">
<INPUT TYPE="button" NAME="button" Value="Test" onClick="testSelect(this.form)">
<SELECT NAME="list" SIZE="3">
<OPTION>This is item 1
<OPTION>This is item 2
<OPTION>This is item 3
</SELECT>
</FORM>
</BODY>
</HTML>
```



# More on Lists

- If you want the text of the selected list item instead of the index, use the following code in the testSelect function:

```
function testSelect (form) {
  Item = form.list.selectedIndex;
  Result = form.list.options[Item].text;
  alert (Result);
}
```

# Other events within a form

- the onclick event handler has been used in all of the examples because that's the one you are most likely to use with your forms.
- JavaScript supports a number of other event handlers as well.
- The event handlers used with form object are:
  - *onfocus* -- an event is triggered with a form object gets input focus (the insertion point is clicked there).
  - *onblur* -- an event is triggered with a form object loses input focus (the insertion point is clicked out of there).

# Other events within a form

- *onchange* -- an event is triggered when a new item is selected in a list box. This event is also trigged with a text or text area box loses focus and the contents of the box has changed.
- *onselect* -- an event is triggered when text in a text or text area box is selected.
- *onsubmit* -- an event is triggered when the form is submitted to the server

# Quick Review

- What is  a HTML Form?
- JavaScript and forms