JavaScript and HTML Documents 7

At the end of this lesson, students should be able to:

- Describe the Document Object Model and the Document Object Model hierarchy
- Write JavaScript that use the properties, methods and event handlers associated withthe HTML DOM's objects
- Use HTML DOM event tag attributes to write JavaScript functions that validate form input

1 Introduction

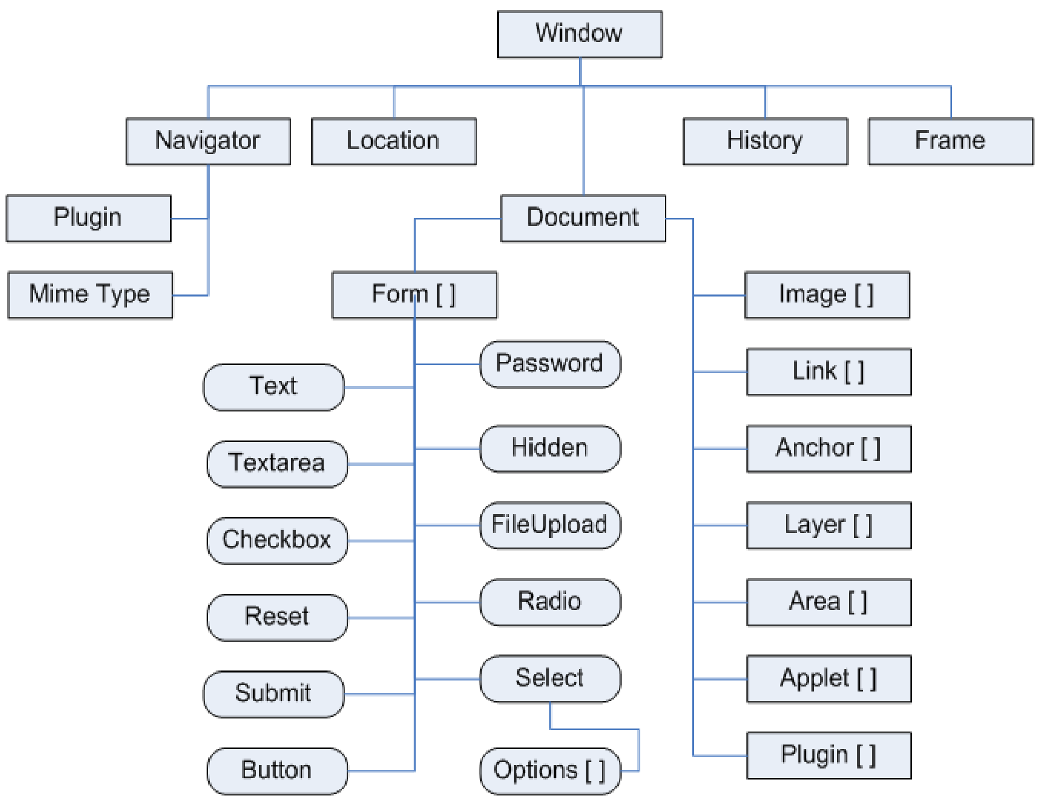
Browser contains built-in objects. You can access these objects, along with their properties and methods, when building the web applications. The hierarchical collection of objects follows a standard referred to as the *Document Object Model* or *DOM*. Using the client-side scripting like JavaScript, programmers can access all of the objects, properties and methods within the DOM. HTML tags are represented in JavaScript as *objects*; tag attributes are represented as *properties.*

An *event* is a notification that something specific has occurred, either with the browser, such as the completion of the loading of a page, or because of a browser user action, such as mouse click on a button. An *event handler* is a script that implicitly executed in response to the appearance of an event. JavaScript defines a collection of events which are associated with HTML tag attributes that can be used to connect to the event handlers. The attributes have names that are closely related to their associated events.

1.1 The Document Object Model Hierarchy

1. The following figure illustrates the entire collection of objects within the DOM and their hierarchical relationship.

The Document Object Model Hierarchy (Kathleen Kalata, 2001)



The window object is the highest level of an object tthat can existin the DOM. Since the window object represents the browser window, the properties and methods associated with it are handled through the window object. The window object contains child objects, properties, methods and events and is the parent to several child objects, including document, frame, location and navigator.

2. The following table is a list of objects and their methods and properties that are commonly used in client-side programming.

Task	Objects with methods and properties
To alter the status message bar	`window.status = “This is the status message”`

To create dialog boxes	`window.alert(“This is the alert message”)`
	`window.confirm(“This is the confirm message”)`
	`window.prompt(“This is the prompt message”, “This is the default string in the textbox”);`

To open a new window and alter its appearance	`window.open(“Mypage.html”, ”window name”,”height=200,width=400, status = no, toolbar=no, menubar=no, location = no”)`
To close a window	`window.close( )`
To display the user agent	`navigator.userAgent`
To display the complete URL of the current document	`location.href`
To redirect the window to a new document	`location.href=(“http://www.web.com”)`
To move to the previous window in the history list	`window.history.back( )`
To move to the next window in the history list	`window.history.next( )`
To write to the document in the current window	`document.write(“String and HTML” + Variables)`
To display the document properties, such as the document title, and last modified date	`document.title` `document.lastModified`
To display form properties, such as the name of the form	`window.document.loginform.name`
To display object properties, such as the name	`document.form1.txtName.name`
To display the object properties, such as the value	`document.form1.txtName.value`
To handle events such as when the object is clicked (onClick)	`<input type =”button” name =`

	“button1” value = “Change Image” onClick = “ChangeImage()”>

1.2 Hands On Exercises

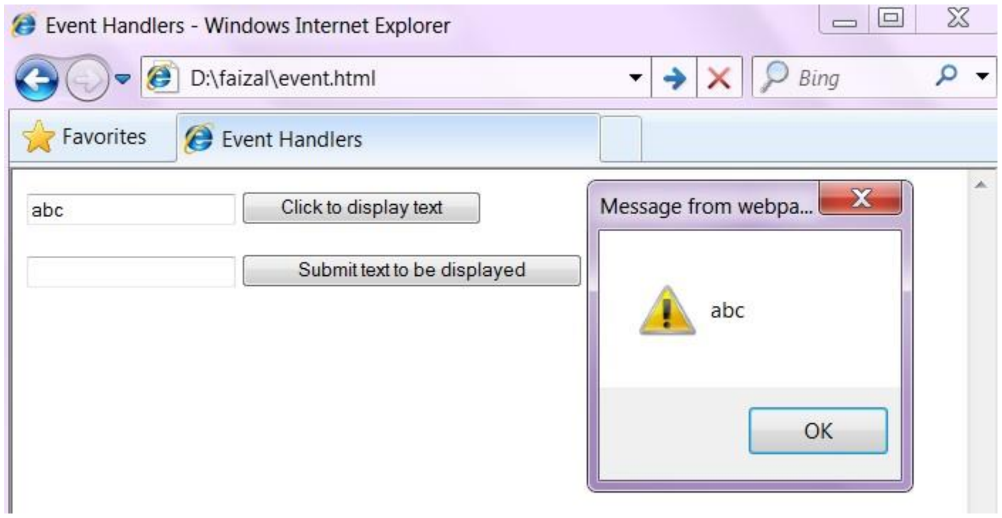
1.2.1 Form Element with onClick and onSubmit event Handlers

1. The following HTML document has two form fields that carry the same action but use different type of event handlers.

```
<html>
<head>
<title>Event Handlers</title>

<script type = "text/javascript">

function DisplayX()
{  alert(document.myform1.x.value);  }

function DisplayY()
{  alert(document.myform2.y.value);  }
</script>

</head>
<body>

<form name="myform1" method="post">
<input name="x" type="text">
<input type="Submit" name="ClickButton" value="Click to  display
text" onclick="DisplayX();">
</form>

<form name="myform2" method="post" onsubmit="DisplayY();">
<input name="y" type="text">
```

```
<input type="Submit" name="SubmitButton" value="Submit text to be
displayed">

</body>
</html>
```

2. When the user enters a value and click the button, the document object property is fetched based on the name of the element form input field accessing it's value. This is called get element by name technique.



3. Another way to access to the property of your document is by using get element by ID technique. Now, modify the JavaScript code in the `DisplayY()` function. You are going to utalize the `getElementById()` method instead.

```
function DisplayY()
{ var thisY = document.getElementById("y");
    alert(thisY.value);   }
```

4. Now, In the HTML form, replace the input field name attribute to ID.

```
<input id="y" type="text" >
```

5. Save your code as **Events.html** and test it in your browser.

Form Field Event Handler and Description
onSubmit – Action performed when the user clicks the Submit button
onSelect – Action performed when the user highlights text
onFocus – Action performed when an object has an attention
onChange – Action performed when the user changes a value and moves off the object
onLoad – Action performed when a document object is loaded

1.2.2 Validation of Form Input

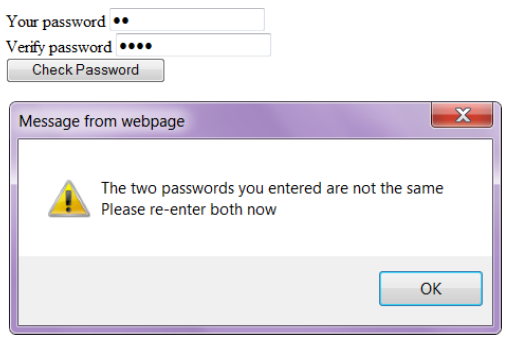
1. Using client side JavaScript is an efficient way to validate the user input in web forms. The browser doesn't have to connect to the server to validate the form, so the user finds out instantly if they've missed out required fields.
2. The code below are segments of JavaScript's function named **'Verify'** that checks if the passwords match when the Submit button is pressed.

```
function Verify()
{
   var pass1 = document.getElementById("pwd1");
   var pass2 = document.getElementById("pwd2");
if (pass1.value != pass2.value)
{
   alert("The two passwords you entered are not the same \n" +
"Please re-enter both now");
   return false;
}
else return true;
}
```

3. The function performs a test is to determine whether the two typed passwords are the same. If they are different, the function calls **`alert`** to generate error message, and returns**`false.`** If they are the same, it returns **`true.`**

4. Create an HTML document and save it as **'Verify.html'** that includes a form consists of *two password input elements*, along with a *Submit* button as shown below. You must name the input elements based on the property in the JavaScript. Use the „`onsubmit`" event handler on `form` object to trigger the call as shown below.

```
<form onsubmit = "Verify();">
Your password <input type = "password" id = "pwd1" /><br />
Verify password <input type = "password"  id = "pwd2"/><br />
<input type = "submit"  name = "submit"  value="Check Password"  />
</form>
```



The two passwords you entered are not the same
Please re-enter both now

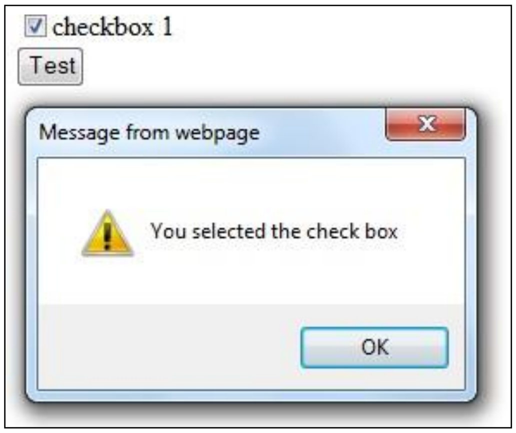
5. Now add more validation scripts in the function to determine whether the user typed the initial password by testing the value of the element against the empty string. If no value has been typed into initial password fields, the function calls alert to produce an error message.

1.2.3 Working with Check Boxes, Radio Buttons Text Areas and Lists

1. In the previous examples, you have learned the text field (`type="text"`) is perhaps the most common form object you'll read (or write) using JavaScript. In addition to text boxes, JavaScript can also be used with:
   - Check box (`type="checkbox"`)
   - Radio button (`type="radio"`)
   - Text area (`<text area>`)
   - Lists (`<select>`)
2. Check boxes are useful for selecting or deselecting a single, independent option while radio buttons are used for making one selection among a group of options. To test if a check box is checked using the checked property, type down this code below and save your file as **MoreFormObjects.html**

```
<html>
<head>
<title>Exploring other Form Objects</title>
<script type ="text/javascript" >
function testbutton (form){
 if (form.check1.checked)
     alert("You selected the check box");
  else
```

```
     alert("You didn't  select the check box");
}
</script>
</head>
</body>
<form name="testform">
<input type="checkbox" name="check1">checkbox 1<br />
<input    type="button"    name="button"    value="test    checkbox"
onclick="testbutton(form);">
</form>
</body>
</html>
```



You selected the check box

3. Add more checkbox tags and test your results. Next, insert these radio buttons objects into your existing form tag.

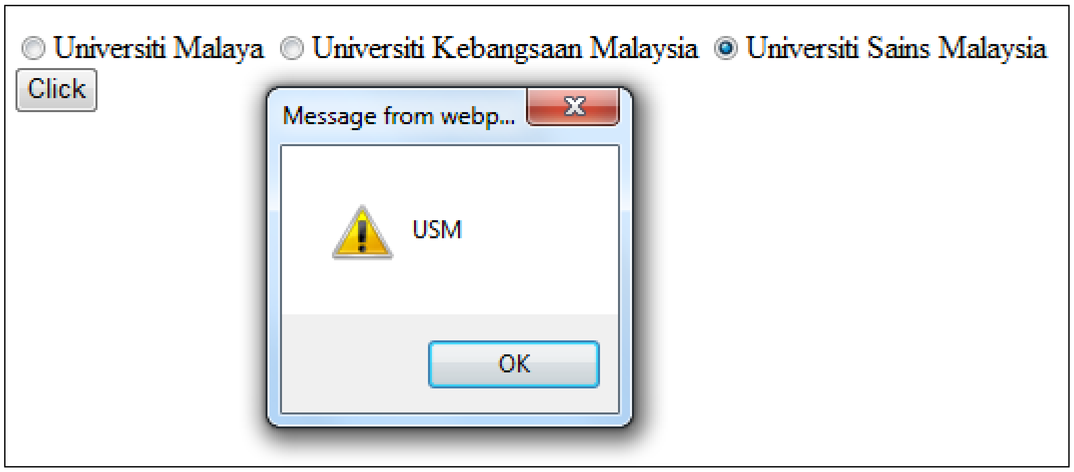
```
<input type = radio name = "rad" value = "UM" >Universiti
 Malaya
<input type = radio name = "rad" value = "UKM" >Universiti
Kebangsaan Malaysia
<input type = radio name = "rad" value = "USM">Universiti Sains
Malaysia<br />
```

4. JavaScript will create an array using the same Radio button object name "rad" group you've provided, then you reference them using the array indexes. Next, insert this loop in the existing **testButton** function.

```
   chosen = "";
```

```
len = document.testform.rad.length;
for (i = 0; i <len; i++)
{ if (document.testform.rad[i].checked)
     { chosen = document.testform.rad[i].value; }
}
       if ((testform.rad[0].checked==false)&&
           (testform.rad[1].checked==false)&&
           (testform.rad[2].checked==false))
          { alert("No university is chosen"); }
  else {  alert(chosen); }
```

5. 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 (starting from 0). Save and view the webpage in your browser.

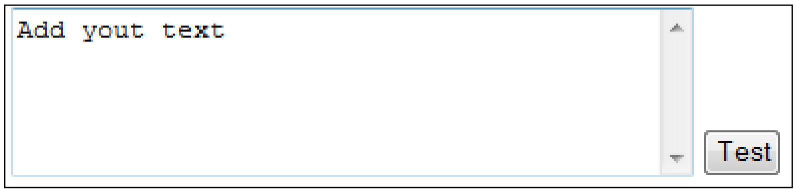


6. Now, type down this code to insert a new text area in your existing form object.

```
<textarea name="myarea" cols="40" rows="5">add your text
</textarea><br />
```

7. Insert alert box that reads the value from the text area object. Place this code in the **testButton** function.

```
alert (form.myarea.value);
```



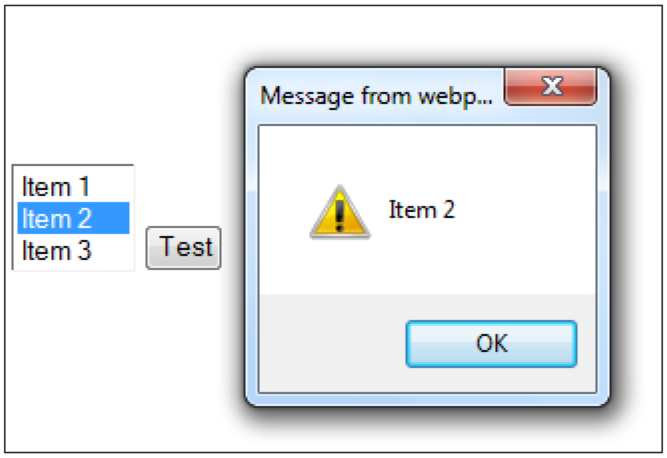
8. Finally, insert a selection list in your form.

```
<select name="list" size="3">
<option>item 1
<option>item 2
<option>item 3
</select><br />
```

9. In your **testButton** function, use the `selectedIndex` property to test which option item is selected in the list. The item is returned as an index value, with 0 being the first option, 1 being the second, and so forth (if no item is selected the value is -1) as shown in the code below.

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

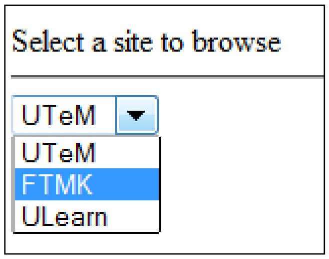
10. Save your file and test the result in your browser.



Item 2

1.2.4 Select Menu with onChange Event Handler

1. The following HTML includes a form that provides a select component, to allow the user to select a Web site from a list of sites. The JavaScript function allows the new website is displayed in the browser when the selection is made.



2. Create a page named **'Location.html'** using text editor. Enter the HTML below. Identify and correct the errors in the script. Save the page, and then view it in your browser.

```
<html> <head>
<title>Jump Menu Demo</title>
<script type = "text/JavaScript">
//Browsing a website using JavaScript
   function Site(val)
   {
   window.location.href = val;
   }
</script>
</head>
<body>
   Select a site to browse <hr />
   <form action = "">
<select name = " SiteSelector" size = "1"
onChange="Site(this.value)">
<option value = "http://www.utem.edu.my"> UTeM </option>
<option value = "http://ftmk.utem.edu.my"> FTMK</option>
<option value = "http://ulearn.utem.edu.my"> ULearn </option>
</select>
</form>
</body></html>
```

3. Save your file and open the webpage in your browser.

1.2.5 Basic Form Validation

Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button. If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process which used to put a lot of burden on the server.

JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.

- **Basic Validation −** First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.
- **Data Format Validation −** Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.

We will take an example to understand the process of validation. Write the following code. Save as simpleform.html.

```
<html>
   <head>
      <title>Form Validation</title>
      <script type = "text/javascript">
         <!--
            // Form validation code will come here.
         //-->
      </script>
   </head>

   <body>
      <form action = "/cgi-bin/test.cgi" name = "myForm" onsubmit =
"return(validate());">
         <table cellspacing = "2" cellpadding = "2" border = "1">

            <tr>
               <td align = "right">Name</td>
               <td><input type = "text" name = "Name" /></td>
            </tr>

            <tr>
               <td align = "right">EMail</td>
               <td><input type = "text" name = "EMail" /></td>
            </tr>

            <tr>
               <td align = "right">Zip Code</td>
               <td><input type = "text" name = "Zip" /></td>
            </tr>

            <tr>
               <td align = "right">Country</td>
               <td>
                  <select name = "Country">
                     <option value = "-1" selected>[choose 
yours]</option>
                     <option value = "1">USA</option>
                     <option value = "2">UK</option>
                     <option value = "3">INDIA</option>
                  </select>
               </td>
            </tr>

            <tr>
               <td align = "right"></td>
               <td><input type = "submit" value = "Submit" /></td>
            </tr>

         </table>
      </form>
   </body>
</html>
```

Basic Form Validation

First let us see how to do a basic form validation. In the above form, we are calling **validate()** to validate data when **onsubmit** event is occurring. Key-in the following code that shows the implementation of this validate() function.

```
<script type = "text/javascript">
   <!--
      // Form validation code will come here.
      function validate() {

         if( document.myForm.Name.value == "" ) {
            alert( "Please provide your name!" );
            document.myForm.Name.focus() ;
            return false;
         }
         if( document.myForm.EMail.value == "" ) {
            alert( "Please provide your Email!" );
            document.myForm.EMail.focus() ;
            return false;
         }
         if( document.myForm.Zip.value == "" || isNaN(
document.myForm.Zip.value ) ||
            document.myForm.Zip.value.length != 5 ) {

            alert( "Please provide a zip in the format #####." );
            document.myForm.Zip.focus() ;
            return false;
         }
         if( document.myForm.Country.value == "-1" ) {
            alert( "Please provide your country!" );
            return false;
         }
         return( true );
      }
   //-->
</script>
```

Data Format Validation

Now we will see how we can validate our entered form data before submitting it to the web server.

The following example shows how to validate an entered email address. An email address must contain at least a '@' sign and a dot (.). Also, the '@' must not be the first character of the email address, and the last dot must at least be one character after the '@' sign.

Example

Try the following code for email validation.

```
<script type = "text/javascript">
   <!--
      function validateEmail() {
         var emailID = document.myForm.EMail.value;
         atpos = emailID.indexOf("@");
         dotpos = emailID.lastIndexOf(".");

         if (atpos < 1 || ( dotpos - atpos < 2 )) {
            alert("Please enter correct email ID")
            document.myForm.EMail.focus() ;
            return false;
         }
         return( true );
      }
   //-->
</script>
```