JavaScript

JavaScript Objects

Agenda

1 JavaScript Objects

Objectives

At the end of this module you will be able to:

- Create Windows & Dialog Boxes
- Use the JavaScript's in-built objects in a web page
- Write code that does Event Handling in HTML pages
- Manipulate HTML Forms through JavaScript dynamically

JavaScript Objects




Objects

An object is a self-contained unit of code having the following characteristics:

- Properties
- Methods
- Identity

Objects

- Like variables, objects can store data - but they can store more pieces of data at once.
- The items of data stored in an object are called the Properties of the object.
- E.g: A person object might include **Bob.address** and **Bob.phone**, where **Bob** is the object and address & phone are its properties.
- Objects have methods or functions which work with the object's data.
- E.g: A person object might include a **display()** method to display the person's information.

Objects: The Window Object

At the top of the browser hierarchy is the window object, which represents a browser window

The properties/methods of this object are:

status

alert()

confirm()

prompt()

Objects: The Window Object

- The ***window.status*** is a property which can be used to change the contents of the browser's status line.
- The ***window.alert()*** method can be used to alert the user in case of closing the window before saving the document. It appears as a dialog box with a exclamatory sign.
- The ***window.confirm()*** method can be used to prompt the user to ask for clarification. It opens a dialog box with a question mark and two buttons with ok and cancel as options.
- The ***window.prompt()*** method can be used to prompt the user in case of accessing any data from the user, such as user name and password. It opens a dialog box with a text box to accept data from the user.

Objects: The Window Object(Contd.).

Simple Example:

```
<SCRIPT LANGUAGE="JavaScript">
function hello(){
      window.alert("You are about to view the Transaction Details
");
 }
</SCRIPT>
<P> Click <A onClick = "hello();" HREF = "transaction-
statement.html">MiniStatement </A>
</P>
```

Objects: The Document Object

- The Document Object represents characteristics of the current HTML page.
- Some of its properties are:
  - Title
  - lastModified
  - fgColor
  - bgColor
- One of its most important method is :
  - write()
- In the browser object hierarchy, the document object is contained in a window object (for a page without frames)

Objects: The Document Object (Contd.).

- The properties of the document object represent characteristics of the current HTML page. Many of these are specified in the <BODY> tag of the document while some are set by the browser when the document is loaded.
- The title property lists the title of the current page, defined by the HTML <TITLE> tag.
- The last modified property is the date the document was last modified.

Example:

```
<HEAD>
<TITLE> Test JavaScript </TITLE>
<SCRIPT Language="JavaScript">
        document.write(document.title+" <BR>");
        document.write(location.toString());
</SCRIPT>
</HEAD>
```

Objects: The Form Object

- The Form object represents an HTML Form. It has the same name as the NAME attribute in the FORM tag
- In the browser object hierarchy, the form object is contained in the document object
- For example, if the first form in a document has the name **form1**, you can refer to it in one of two ways:
  - *document.form1*
  - *document.forms[0]*

Objects: Frame Objects

- Each Frame in a frame set is represented as a frame object.
- A frame set contains an array of frame objects representing all the frames in it.

You can refer to a particular frame:

- By name - if it has one
- By reference to the parent and the array index of that frame

Objects: Frame Objects(Contd.).

- JavaScript allows you to manipulate frames and contents of frames by providing objects that represent the frames in the current frameset.

- The frames are either available as objects with the same name as the frame name or as an array of frames.

- To access one frame from another frame in the same frameset, you need to use the 'parent' frame as a reference through which the other frames can be accessed. If there are multiple levels of nested frames, the 'top' frame can be used as the reference to access other frames.

Objects: The Math Object

- The Math object can't be created, since it exists automatically in all Java Script Programs.
- Its properties represent mathematical constants.

  Eg:Math.PI
- Its methods are mathematical functions.
- Three of the most useful methods of the Math object enable you to round decimal values up and down:
  - *Math.ceil()* rounds a number up to the next integer.
  - *Math.floor()* rounds a number down to the next integer.
  - *Math.round()* rounds a number to the nearest integer.

- All these take a single parameter: the number to be rounded. You might notice one thing missing: the ability to round to a decimal place, such as for dollar amounts. You can easily simulate this, though.

Math methods

abs(x)	Returns absolute value of x
ceil(x)	Returns the smallest integer greater than or equal to x (round up)
cos(x)	Returns cosine of x, where x is in radians
floor(x)	Returns the largest integer less than or equal to x (round down)
log(x)	Returns the natural logarithm (base E) of x
max(a, b)	Returns the larger of a and b
min(a, b)	Returns the lesser of a and b
random()	Returns a pseudorandom number between 0 and 1
round(x) -	Rounds x up or down to the nearest integer. It rounds .5 up

What will this code print ?

```
<html>
<script language = "JavaScript">
var x=20;
var y=7;
var a1= Math.ceil(x/y);
var a2 = Math.floor(x/y);
var a3 = Math.round(x/y);
document.write("ceil = "+a1+"<BR>");
document.write("floor = "+a2+"<BR>");
document.write("round = "+a3+"<BR>");
</script>
</body>
</html>
```

Objects: The String Object

- Any String variable in JavaScript is a String object.
- String object have a single property, length.

Concatenating strings (+)

indexOf

lastIndexOf

charAt

length

split

substring

substr

toLowerCase and toUpperCase

Objects: The String Object

- You can perform string functions using a variety of methods.
- Methods enable you to convert strings, work with pieces of strings, search for values, control the string's appearance on the HTML page, and control HTML links and anchors.
- String Conversions methods:
  - toUpperCase() converts all characters in the string to uppercase.
  - toLowerCase() converts all characters in the string to lowercase.

Examples:

```
<script language="javascript">

document.write("Wipro" + "EC4"); //Concatenation
var b = 'I am a JavaScript hacker.'
document.write(b.lastIndexOf('a'))
document.write(b.lastIndexOf('w'))
document.write(a.substring(4,8)); // extract characters from 4 to 8 location
document.write(a.substr(4,8)); // extract 8 characters from 4 location
</script>
```

What will this code print ?

```
<html><body>
<script language="JavaScript">
var url="http://mywipro.wipro.com";
var x = url.indexOf("wipro");
document.write("The string wipro is located at "+x+" position<BR>");
var y = url.indexOf("o");
var z = url.lastIndexOf("o");
document.write("The first occurence of the character 'o' is at "+y+"
  position<BR>");
document.write("The last  occurence of the character 'o' is at "+z+"
  position<BR>");
var a = url.charAt(23);
document.write("The character at the position 23 is " +a);
</script>
</body></html>
```

What will this code print ? (Contd.).

```
<script language="javascript">
var x= "zero one two three four";
var SplitResult = x.split(" ");

for(i = 0; i < SplitResult.length; i++){
    document.write("<br /> Element " + i + " = " +
  SplitResult[i]);
}
</script>
```

Output :

Element 0 = zero
Element 1 = one
Element 2 = two
Element 3 = three
Element 4 = four

What will this code print ? (Contd.).

```
<html>
<body>
<script language="JavaScript">
var x;
alpha="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
x=alpha.substring(7,12);
y=alpha.substr(7,12);
document.write(x+"<BR>");
document.write(y);
</script>
</body>
</html>
```

String Object & Comparison Operators

- The following are the different Comparison Operators available in Javascript.

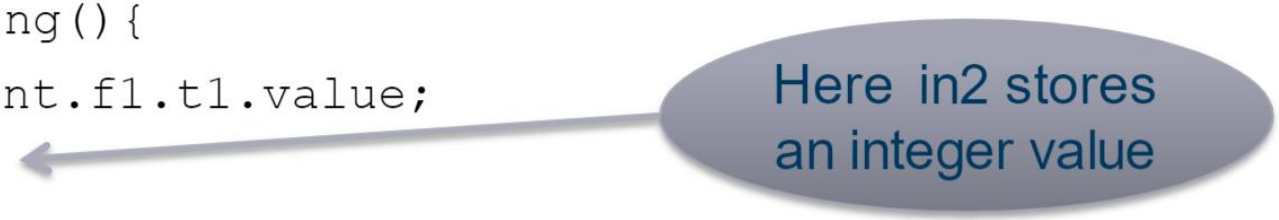
  >, <, >=, <=, !=, == and ===

- While we are aware of the uses of most of these operators, we may not have used "===" till now.
- "==" and "===" are both used to compare the values, but there is a difference.
- "==" represents "is equal to"
- "===" represents "is exactly equal to" (It will not only check whether the values are equal or not but also whether the data type is same or not)

Let us understand the difference through an example given in the next slide.

String Object & Comparison Operators (Contd.).

```
function comparing(){
  var in1=document.f1.t1.value;
  var in2=10;
  if(in1==in2) {
    document.write("Both the values are equal <BR>");
    if(in1===in2)
        document.write("Both the values are of same type");
    else
        document.write("Both the values are of different type");
  }
  else
    document.write("Both the values are unequal");
}
```



String Object & Comparison Operators (Contd.).

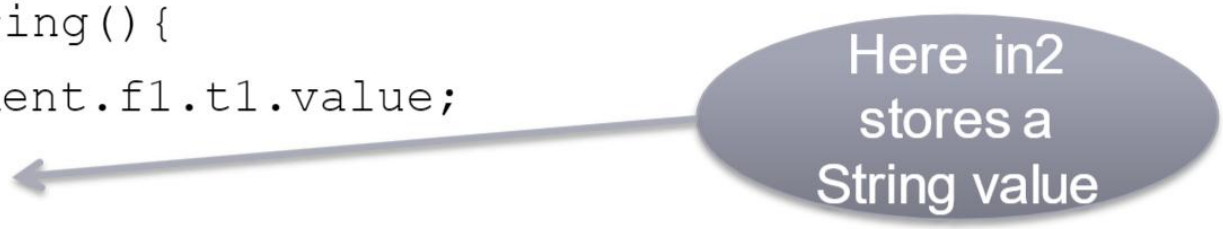
```
<form name="f1"> Enter only digits :
<input type="text" name="t1" />
<input type = button value = "Click here" onClick="comparing()" />
</form>
```

Output :

Both the values are equal
Both the values are of different type

String Object & Comparison Operators (Contd.).

```
function comparing(){
  var in1=document.f1.t1.value;
  var in2="10";
  if(in1==in2) {
    document.write("Both the values are equal <BR>");
    if(in1===in2)
        document.write("Both the values are of same type");
    else
        document.write("Both the values are of different type");
  }
  else
    document.write("Both the values are unequal");
}
```



String Object & Comparison Operators (Contd.).

```
<form name="f1"> Enter only digits :
<input type="text" name="t1" />
<input type = button value = "Click here" onClick="comparing()" />
</form>
```

Output :

Both the values are equal
Both the values are of same type

Objects: The Date Object

- The Date object is a built-in JavaScript object that enables you to conveniently work with dates and times.
- You can create a Date object any time you need to store a date, and use the Date object's methods to work with the date.
- You can create a Date object using the new keyword.
- You can use any of the following formats:

```
birthday = new Date();
birthday = new Date("June 20, 1996 08:00:00");
birthday = new Date(6, 20, 96);
birthday = new Date(6, 20, 96, 8, 0, 0);
```

Some Important Keywords

- The ***"this"*** Keyword

  "this" is used to refer to the current object

- The ***"with"*** Keyword

  You can use it to make JavaScript programming easier-or at least easier to type

  The with keyword specifies an object and it is followed by a set of statements enclosed in braces

Some Important Keywords(Contd.).

```
<html>
<script language = "JavaScript">
    function tot() {
        with(document.Simple) {
            var x=parseInt(first.value)
            var y=parseInt(elements[1].value)
            var z=parseInt(third.value)
            sum.value = x+y+z
        }
    }
</script>
```

Some Important Keywords(Contd.).

```
<form name = "Simple">
<input type = "text" size = "4" name = first value = " "> <BR>
<input type = text size = 4 name = second value = " "> <BR>
<input type = text size = 4 name = third value = " "> <BR>
<input type = button  value = "Click" onClick = "tot();"> <BR>
<input type = text  name = sum value = " "> <BR>
</form>
</html>
```

Arrays

- Arrays need to be created using new keyword
- Once you define the array, you can access its elements by using brackets to indicate the index
- For example, the statement to create an array called scores with 20 values is as follows: scores= new Array(20)
- Array Object Methods :
  - *join()*
  - *reverse()*
  - *sort()*
- Once you define the array, you can access its elements by using brackets to indicate the index . As an example, this statement creates an array called scores with 20 values:

  Scores = new Array(20);

Arrays(Contd.).

Array object methods:

- ***join()*** quickly joins all the array's elements, resulting in a string. The elements are separated by commas by default.
- ***reverse()*** returns a reversed version of the array: the last element becomes the first, and the first element becomes the last.
- ***sort()*** returns a sorted version of the array.

```
<script language="javascript">
var Name=new Array(2);
Name[0]="Raghu";
Name[1]="Kiran"
var s=new Array("Saab","Volvo","BMW");
for( var i=0;i<s.length;i++)
document.writeln(s[i]);
for( var k=0;k<Name.length;k++)
document.writeln(Name[k]);
</script>
```

Events

- Events refer to things that happen to the browser and is used to trigger portions of program.
- Events pertain to the web page containing the script.
- When the user clicks on a link, selects or enters text, or even moves the mouse over part of the page, an event occurs.
- You can use JavaScript to respond to these events. For example, you can have custom messages displayed in the status line (or somewhere else on the page) as the user moves the mouse over links. You can also update fields in a form whenever another field changes.

Events(Contd.).

- **onBlur** Occurs when an object on the page loses focus
- **onChange** Occurs when a text field is changed by the user
- **onClick** Occurs when the user clicks on an item
- **onFocus** Occurs when an item gains focus
- **onLoad** Occurs when the page (or an image) finishes loading
- **onMouseOver** Occurs when the mouse pointer moves over an item
- **onMouseOut** Occurs when the mouse pointer moves off an item
- **onSelect** Occurs when the user selects text in a text area
- **onSubmit** Occurs when a submit button is pressed
- **onUnload** Occurs when the user leaves the document or exits

Events: Event Handlers

- Embedded in **html** tags, typically as part of forms, but are also included as a part of some anchors and links

- Virtually anything a user can do to interact with a page is covered with the event handlers, from moving the mouse to leaving the current page

Time Outs

- Time outs refer to Statements that will be executed after a certain amount of time elapses.

- Handy for periodically updating a Web Page or for delaying the display of a message or execution of a function.
- You begin a timeout with the ***setTimeout()*** Method.
- Before a timeout has elapsed, you can stop it with the ***clearTimeout()*** method, specifying the identifier of the timeout to stop.

Time Outs(Contd.).

```
// Update a page every 2 seconds
    var counter = 0;
// call Update function in 2 seconds after first load
    ID=window.setTimeout("Update();",2000);
    function Update(){
    counter++;
    window.status="The counter is now at "+counter;
document.form1.input1.value="The counter is now at "+counter;
//set another timeout for the next count
ID=window.setTimeout("Update();",2000);}
<FORM NAME="form1">
<INPUT TYPE="text" NAME="input1" SIZE="40"><BR><INPUT TYPE="button"
VALUE="RESET" onClick="counter=0;"><BR>
<INPUT TYPE="button" VALUE="STOP" onClick="window.clearTimeout(ID);">
    </FORM>
```

Creating Windows

- Opening new browser windows is a great feature of JavaScript.
- You can either load a new document (for example a HTML-document) to the new window or you can create new documents (on-the-fly).
- Here is a list of the properties a window can have:
  - directories yes|no
  - height *number of pixels*
  - location yes|no
  - menubar yes|no
  - resizable yes|no
  - scrollbars yes|no
  - status yes|no
  - toolbar yes|no
  - width *number of pixels*

Example to create New Window

Example to create New Window

```
<html>
<head>
<script language="JavaScript">
<!-- hide
function openWin3() {
  myWin= open("", "displayWindow",

"width=500,height=400,status=yes,toolbar=yes,menubar=yes")
;
  // open document for further output
  myWin.document.open();
  // create document
  myWin.document.write("<html><head><title>On-the-fly");
  myWin.document.write("</title></head><body>");
  myWin.document.write("<center><font size=+3>");
  myWin.document.write("This HTML-document has been
created ");
  myWin.document.write("with the help of JavaScript!");
myWin.document.write("</font></center>");
  myWin.document.write("</body></html>");

  // close the document - (not the window!)
  myWin.document.close();
}
// -->
</script>
</head>
<body>
<form>
<input type=button value="On-the-fly"
onClick="openWin3()">
</form>
</body>
</html>
```

Quiz

What happens when you move the mouse pointer on the image displayed?

```
<html>
 <body bgcolor="pink">
<img name=ash src="ash2.bmp" onmousemove=vh() onmouseout=hv()>
<script>
function vh() {
  document.ash.src="ash3.bmp";
}
function hv(){
  document.ash.src="ash2.bmp";
}
</script>
</body></html>
```

Quiz (Contd.).

What happens when you move the mouse pointer on the image displayed? What happens when you move the mouse pointer out of the image?

```
<html>
<img name=ash src="ash2.bmp" onmousemove=vh() onmouseout=hv()>
<script>
function vh(){
  document.ash.height=document.ash.height+1;
  document.ash.width=document.ash.width+1;
}
function hv(){
  document.ash.height=document.ash.height-100;
  document.ash.width=document.ash.width-100;
}
</script></body></html>
```

Summary

In this module, you were able to:

- Create Windows & Dialog Boxes
- Use the JavaScript's in-built objects in a web page
- Write code that does Event Handling in HTML pages
- Manipulate HTML Forms through JavaScript dynamically

Thank You