Lecture 5

This lecture, will learn about

1. Browser Objects
2. Window Object
3. Document Object
4. Location Object
5. Navigator Object
6. History Object
7. Screen Object
8. Events

Not only is Javascript object-based, but the browser is also made up of objects. When Javascript is running in the browser, we can access the browser's objects in exactly the same way we used Javascript's native objects in the last chapter. But what kinds of objects does the browser provide for us to use?

The browser makes available to us number of objects. For example, there is a window object corresponding to the window of the browser. We have already been using two methods of this object, namely the alert() and prompt() methods. For simplicity we previously called these functions, but they are in fact methods of the browser's window object

Another object made available by the browser is the page itself, represented by the document object. Again, we have already used methods and properties of this object. We have also been using the write() method of the document object to write information to the page.

A variety of other objects exist, representing a lot of the HTML that we write in the page. For example, there is an img object for each <img> tag that we use to insert an image into our document.

The collection of objects that the browser makes available to us for use with Javascript is generally called the *Browser Object Model (BOM)*. Some books refer to BOM as DOM (Document Object Model).

But before starting with BOM, I would like to introduce events.

Connecting Code to Web Page Events

What are events?

Events occur when something in particular happens or when user initiates some thing on the browser. For example:

1. User clicking on a page – This is onClick event.
2. User clicking on a hyper link – This is onClick event
3. Moving mouse pointer over some text – This is onMouseOver event
4. Moving mouse pointer out of some text – This is onMouseOut event.

5. When you write a URL on address bar of browser and click enter, that loads a page – There is an event associated with loading of page called as onLoad.
6. When you leave a web page or close browser – There is an event associated with closing browser or leaving a web page called as onUnLoad.

These were some of the most common used events. Take a real life example: We want to make menu pop up after we take mouse over some menu item.

In the above scenario, taking the mouse over a menu item is a onMouseOver event. We can associate a function say f1()with onMouseOver event. Whenever user will take his mouse over menu item, f1() will be called. Obviously f1() will display popmenu.

Let's create a simple HTML page with a single hyperlink, given by the element <A>. Associated to this element is the Anchor object. One of the events the Anchor object has is the click event.

```
<html>
<body>
<A href="somepage.htm" name="linkSomePage">
      Click Me
</A>
</body>
</html>
```

When I click on hyperlink “Click me”, page somepage.htm is loaded on the browser. Now, if I want I can associate an event with this anchor tag. Whenever user clicks his mouse on “Click me”, an event is fired. Actually even in the above example, event is fired, but we havenot written the code to capture it.

```
<A href="somepage.htm" name="linkSomePage" onClick = “capture_click()”> Click Me </A>
```

Above line means, when user clicks his mouse on “Click Me”, 2 things happen

1. onClick event is fired, which is also captured and hence function capture_click() is called.
2. Load the page somepage.html.

Below is a complete program for capturing onClick event associated with <a>.

```
<html>
<head>
        <script language="JavaScript">
                function capture_click()
                {
                        alert("User clicked his mouse on Click Me");
                }
        </script>
</head>
<body>
<A href="somepage.htm" name="linkSomePage" onClick="capture_click()">
        Click Me
</A>
```

```
</body>
</html>
```

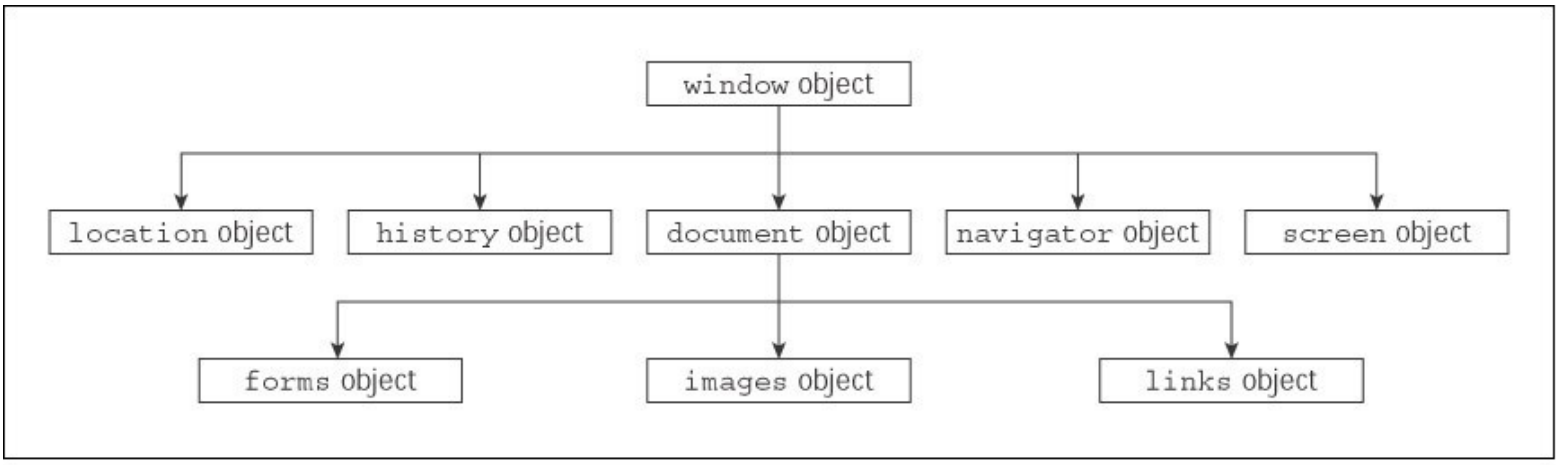
I can also associate onMouseOver event with the above anchor tag. I just need to write onMouseOver instead of OnClick.

```
<html>
<head>
	<script language="JavaScript">
		function capture_Mouseover()
		{
			alert("mouse over me");
		}
	</script>
</head>
<body>
<A href="somepage.htm" name="linkSomePage" onMouseOver="capture_Mouseover()">
	Click Me
</A>
</body>
</html>
```

Later we will see use of onMouseOut event with images object. We will also see usage of onLoad and onUnLoad event.

Introduction to Browser Objects

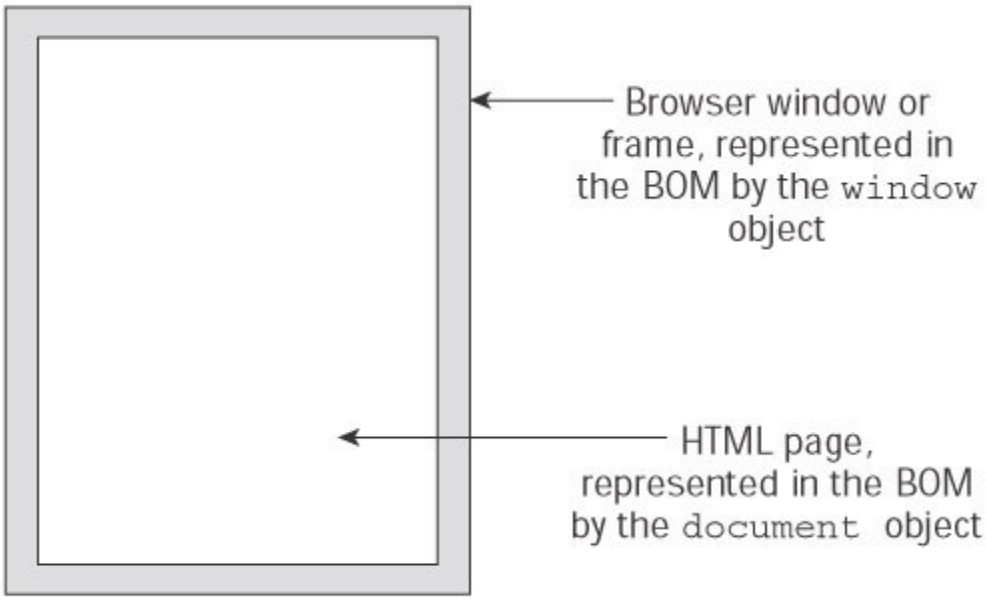
When Javascript is running in a web page, it has access to a large number of other objects made available by the web browser. Unlike native objects, we do not need to create browser objects. They are available to use through the browser.



The BOM has a hierarchy. At the very top of this hierarchy is the window object. You can think of this as representing the frame of the browser and everything associated with it, such as the

scrollbars, navigator bar icons, and so on.

Contained inside our window frame is the page. The page is represented in the BOM by the document object. You can see this in the figure below.



- If we want to make some change in the window like width, height etc, we will use window object (We will learn this later).
- If we want to make some change in page like background color, text color, images etc, we will use document object.

The Window Object

The window object represents the browser's frame or window in which your web page is contained

Below is the list of some of the things that we can do using the properties of Window Object.

1. We can find what browser is running.
2. The pages the user has visited.
3. The size of the browser window.
4. The size of the user's screen, and much more.
5. To access and change the text in the browser's status bar.
6. Change the page that is loaded.
7. Open new windows.

The above list is not complete, I have listed few things that you can do using properties available through Window Object.

The window object is a *global object*, which means we don't need to use its name to access its properties and methods. For example, the alert() function we have been using since the beginning is, in fact, the alert() method of the window object. Although we have been using this simply as

```
alert("Hello!");
```

We could write

```
window.alert("Hello!");
```

However, since the window object is the global object, it is perfectly correct to use the first version.

Some of the properties of the window object are themselves objects. These are

1. document – represents our page.
2. navigator – holds information about the browser, like type and version of browser.
3. history – contains the name of pages visited by user.
4. Screen – contains information about the size of screen like width, height, resolution etc.
5. location – contains the name and location of currently open web page.

To display a message in the window's status bar, we need to use the window object's defaultStatus property. To do this we can write

```
window.defaultStatus = "Hello and Welcome";
```

or, because the window is the global object, we can just write

```
defaultStatus = "Hello and Welcome";
```

Here is a program for the same.

```
<html>
<head>
<script language="JavaScript" type="text/JavaScript">
window.defaultStatus = "Hello and Welcome";
</script>
</head>
</html>
```

The History Object

The history object keeps track of each page that the user visits. It enables the user to click the browser's Back and Forward buttons to revisit pages. We have access to this object via the window object's history property.

We can either use window.history or simply history.

- Like the native JavaScript Array object, the history object has a length property. This can be used to find out how many pages are in the history array.
- The history object has the back() and forward() methods. When they are called, the location of the page currently loaded in the browser is changed to the previous or next page that the user has visited.
- The history object also has the go() method. This takes one parameter that specifies how far forward or backward in the history stack you want to go. For example, if you wanted to return the user to the page before the previous page, you'd write

```
history.go(-2);
```

To go forward three pages, you'd write

```
history.go(3);.
```

Note that go(-1) and back() are equivalent, as are go(1) and forward().

Below is a program that demonstrates use of history object.

```
<html>
<head>
<script language="JavaScript" type="text/JavaScript">

        function goBack()
        {
                alert("back");history.back();
        }
```

```
        function goNext()
        {
                alert("forward");history.forward();
        }
</script>
</head>
<body>
<a onClick = goBack()> Back</a> <br>
<a onClick = goNext()> Next </a>
</body>
</html>
```

The location Object

The location object contains lots of useful information about the current page's location. It contain the URL (Uniform Resource Locator) for the page, but also the server hosting the page, the port number of the server connection, and the protocol used. This information is made available through the location object's href, hostname, port, and protocol properties.

1. location.href – returns the URL of the page that is currently open in the browser
2. location.hostname – returns the hostname part of URL that is currently open in the browser.
3. location.port – returns the port number, http by default uses port 8080.
4. location.protocol – returns the protocol used for accessing internet, we currently use HTTP or ftp etc.

Below is a program that returns the URL that is currently open in the browser. Then user is prompted with a question to enter a new URL. Our program will receive user answer and open a new URL.

```
<html>
<head>
<script language="JavaScript" type="text/JavaScript">

        function getURL()
        {
                aURL = window.location.href;
                alert("URL of your browser is " + aURL);
                uURL=window.prompt("Enter the url of the page that you want to open");
                window.location.href = "http://" + uURL;
        }



</script>
</head>
<body>
<a onClick = goBack()> Back</a> <br>
<a onClick = goNext()> Next </a>
</body>
</html>
```

The Screen Object

The screen object property of the window object contains a lot of information about the display capabilities of the client machine. Its properties include the height and width properties, which indicate the vertical and horizontal range of the screen in pixels. Another property of the screen object, is the colorDepth property. This tells us the number of bits used for colors on the client's screen.

1. window.screen.width – It returns the width of user computer screen.

2. Window.screen.height- It returns the height of user computer screen.
3. Window.screen.colorDepth – It returns the number of bits used for each pixel. It returns a value of 1, 4, 8, 15, 16, 24, or 32. 1 means, each pixel on your screen is composed of 1 bit and can display one of the $2^1 = 2$ colors. Similarly, if 8 is returned, that means, each pixel on your screen is composed of 8 bits and can display one of the $2^8 = 256$ colors

```
<html>
<head>
<script language="JavaScript" type="text/JavaScript">

	function getWindowProperties()
	{
		windowProperties = "Width of browser is "+ screen.width + " pixels";
		windowProperties += ", Height of browser is " + screen.height + " pixels";
		windowProperties += " and Color depth of screen is" + screen.colorDepth + " bits";

		alert(windowProperties);
	}


</script>
</head>
<body>
<a onClick = "getWindowProperties()"> Get Window Properties</a> <br>
</body>
</html>
```

Using the document Object

We've already come across some of the document object's properties and methods, for example the write() method. We will be learning more about document Object. We can do a lot with document Object, I will start with a very simple example of document object i.e. Setting the background color and gradually we will learn more about using document object.

Below is a program which sets the background color of the page using bgColor property of document Object. By default background color of web page is white. We will provide three color options to user pink, skyblue, yellow. User clicks on the color and the screen background color changes.

```
<html>
<head>
<script language="JavaScript" type="text/JavaScript">

	function setColor(colo)
	{
		switch(colo)
		{
			case 1 : document.bgColor = "pink";break;
			case 2 : document.bgColor = "skyblue";break;
			case 3 : document.bgColor = "yellow";break;
		}
	}
</script>
</head>
<body>
Change Colo:    <a onClick = "setColor(1)" href="#"> pink</a>  <a onClick = "setColor(2)"
href="#"> skyblue</a>  <a onClick = "setColor(3)" href="#"> yellow</a>
</body>
</html>
```

We have used document.bgColor property to set the background color of web page.

Now let's look at some of the slightly more complex properties of the document object. These properties have in common the fact that they all contain arrays. We will be discussing images, and links array. Images and Links array are properties of document object.

The images Array

As we know, we can insert an image into an HTML page using the following tag:

```
<img ALT="USA" name=myImage src="usa.gif">
```

The browser makes this image available for us to script in JavaScript by creating an img object for it with the name myImage. In fact, each image on your page has an img object created for it.

Each of the img objects in a page is stored in the images[] array. This array is a property of the document object. The first image on the page is found in the element document.images[0], the second in document.images[1], and so on

```
<HTML>
<HEAD>
<TITLE> Open a New Document </TITLE>
<script language="JavaScript">
        function display_wandh()
        {
                window.alert(document.images.length);
                n=document.images.length
                for(i=0;i<n;i++)
                        window.alert("name="+document.images[i].name+", width="+document.images[i].width+",
height="+document.images[i].height);
        }
</script>
</HEAD>
        <img src="book.gif" name="book" width=100 height=100>  <br>
        <img src="music.gif" name="music" width=200 height=300> <br>
        <img src="soccer.gif" name="soccer" width=100 height=200> <br>

<BODY>
<a href="#" onClick="display_wandh()"> Display Width and Heigh of all the image </a> <br>
</BODY>
</HTML>
```

In the following code, as you move the mouse over the image, image changes, and as soon as you take the mouse out of the image, same image appears again.

```
<HTML>
<HEAD>
<TITLE> Open a New Document </TITLE>
```

```
<script language="JavaScript">
	function changeImage()
	{
		book.src="music.gif";
	}

	function reinstate()
	{
		book.src="book.gif";
	}
</script>
</HEAD>
	<img src="book.gif" name="book" width=100 height=100 onMouseOver="changeImage()"
onMouseOut="reinstate()">
<BODY>
</BODY>
</HTML>
```

Below is a program that shows image randomly. Now, each time you move the mouse over the image, you will see a different image.

```
<HTML>
<HEAD>
<TITLE> Open a New Document </TITLE>
 </HEAD>
<BODY>
<script language="JavaScript">
	 function changeImage()
	{
		var myImage=new Array(4);
		myImage[0] = "soccer.gif";
		myImage[1] = "asia.gif";
		myImage[2] = "music.gif";
		myImage[3] = "book.gif";

		var rand_number = Math.absolute(Math.random() * 4);
		document.image.src=myImage[rand_number];
	}
```

```
</script>
        <img src="book.gif" onMouseOver="changeImage()" onMouseOut="changeImage()" name="image">
</BODY>
</HTML>
```

Using Links Object

We defined an event with anchor tag <a>, using onClick = “name_of_function()”. Like Javascript makes an array of images for each image on web page. Similarly, Javascript also makes an arraty of links for each <a> on web page.

First anchor tag is referred as document.links[0]

second anchor tag is referred as document.links[1] and so on.

```
<html>
<body>
<script language="JavaScript">
function linkSomePage_onclick()
{
   alert('This link is going nowhere');
   return false;
}
</script>

<A href="somepage.htm" name="linkSomePage">
   Click Me
</A>

<script language="JavaScript" type="text/JavaScript">
   window.document.links[0].onclick = linkSomePage_onclick;
</script>
</body>
</html>
```