Program: Information Technology

Course Name: Client-Side Scripting

Course Code: 22519

Course Outcome: Use JavaScript for handling Cookies

Unit Outcomes: a) Create cookies based on given problem.
b) Develop javascript to manage a cookie in the given manner.

Learning Outcome 1 :
Student should understand how cookies are handled.

What we will learn today

- Basics of Cookies
- Reading a Cookie value
- Writing a Cookie value
- Creating a cookie
- Deleting a Cookie
- Setting the Expiration Date of Cookie
- Example

Key takeaways

Concept of cookies

Basic of Cookies

- A cookie is an amount of information that persists between a server-side and a client-side.
- A cookie contains the information as a string generally in the form of a name-value pair separated by semi-colons.
- It maintains the state of a user and remembers the user's information among all the web pages.

Read and write a Cookie with JavaScript

- With JavaScript, cookies can be read like this:

  Syntax: `var x = document.cookie;`

- You can access the cookie like this which will return all the cookies saved for the current domain.

Creating and deleting a Cookie with JavaScript

In JavaScript, we can create, read, update and delete a cookie by using **document.cookie** property.

The following syntax is used to create a cookie:

```
document.cookie="name=value";
```

You can also add an expiry date (in GMT time).

By default, the cookie is deleted when the browser is closed:

```
document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 GMT";
```

With a path parameter, you can tell the browser what path the cookie belongs to.

By default, the cookie belongs to the current page.

```
document.cookie = "username=Chirag Shetty; expires=Thu, 18 Dec 2013 12:00:00 GMT; path=/";
```

Example

```
</head>    <body>
<input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
   <script>
   function setCookie()
   {
      document.cookie="username=Vidyalanakr Polytechnic;expires=Mon, 3 Aug 2020 00:00:00 GMT";
   }
   function getCookie()
   {
      if(document.cookie.length!=0)
      {
         var array=document.cookie.split("=");
      alert("Name="+array[0]+" "+"Value="+array[1]);
      }
      else
      {
      alert("Cookie not available");
      }
   }
   </script>
</body>    </html>
```

Setting the Expiration date of cookie

Cookies are transient by default; the values they store last for the duration of the web browser session but are lost when the user exits the browser.

User can extend the life of a cookie beyond the current browser session by setting an expiration date and saving the expiration date within the cookie.

For example,

```
document.cookie="username=VP; expires=Tues,04 Aug 2020 00:00:00 GMT ";
```

Example

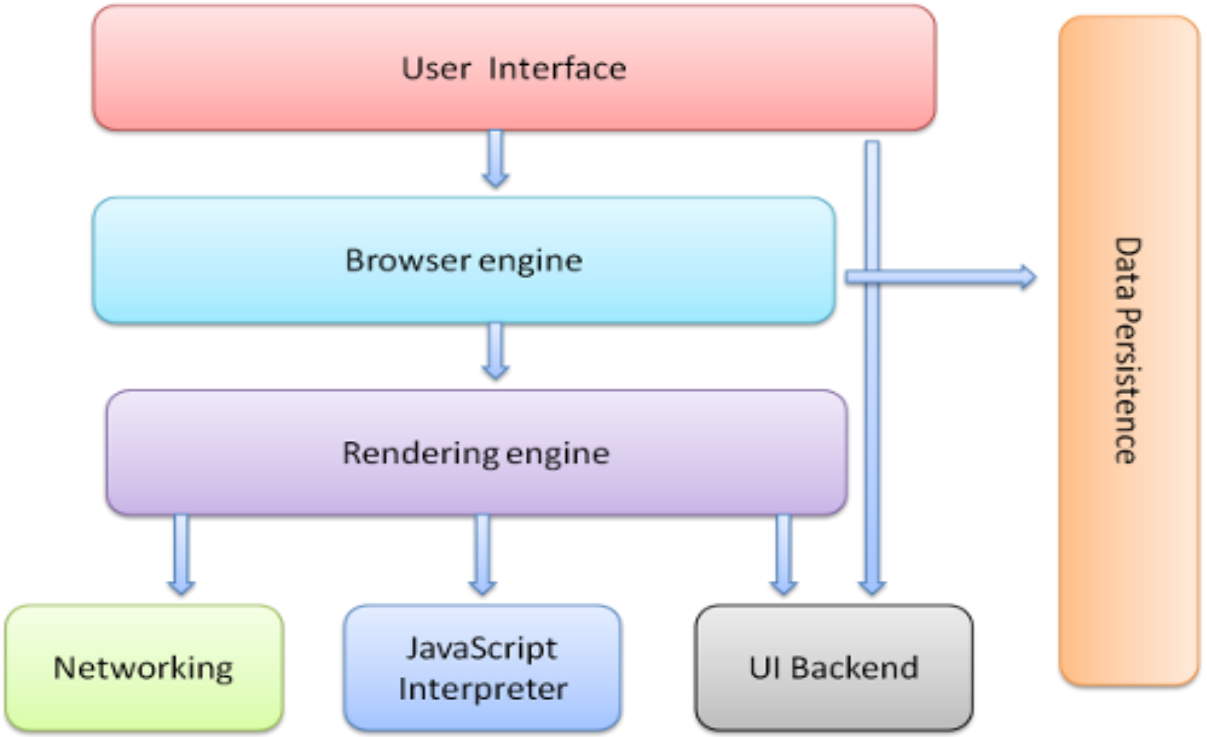
```
<script>
function writeCookie()
{
var d=new Date();
d.setTime(d.getTime()+(1000*60*60*24));
with(document.myform)
{
document.cookie="Name=" + person.value + ";expires=" +d.toGMTString();
}
}
function readCookie()
{
if(document.cookie=="")
document.write("cookies not found");
else
document.write(document.cookie);
}
</script>
```

Learning Outcome 2 :
Student should understand basic concepts of Document Object Model.

Browser

- A web browser (commonly referred to as a browser) is a software application for retrieving, presenting and traversing information resources on the World Wide Web.
- The major web browsers are Firefox, Internet Explorer, Google Chrome, Opera, and Safari.

Components of Browser

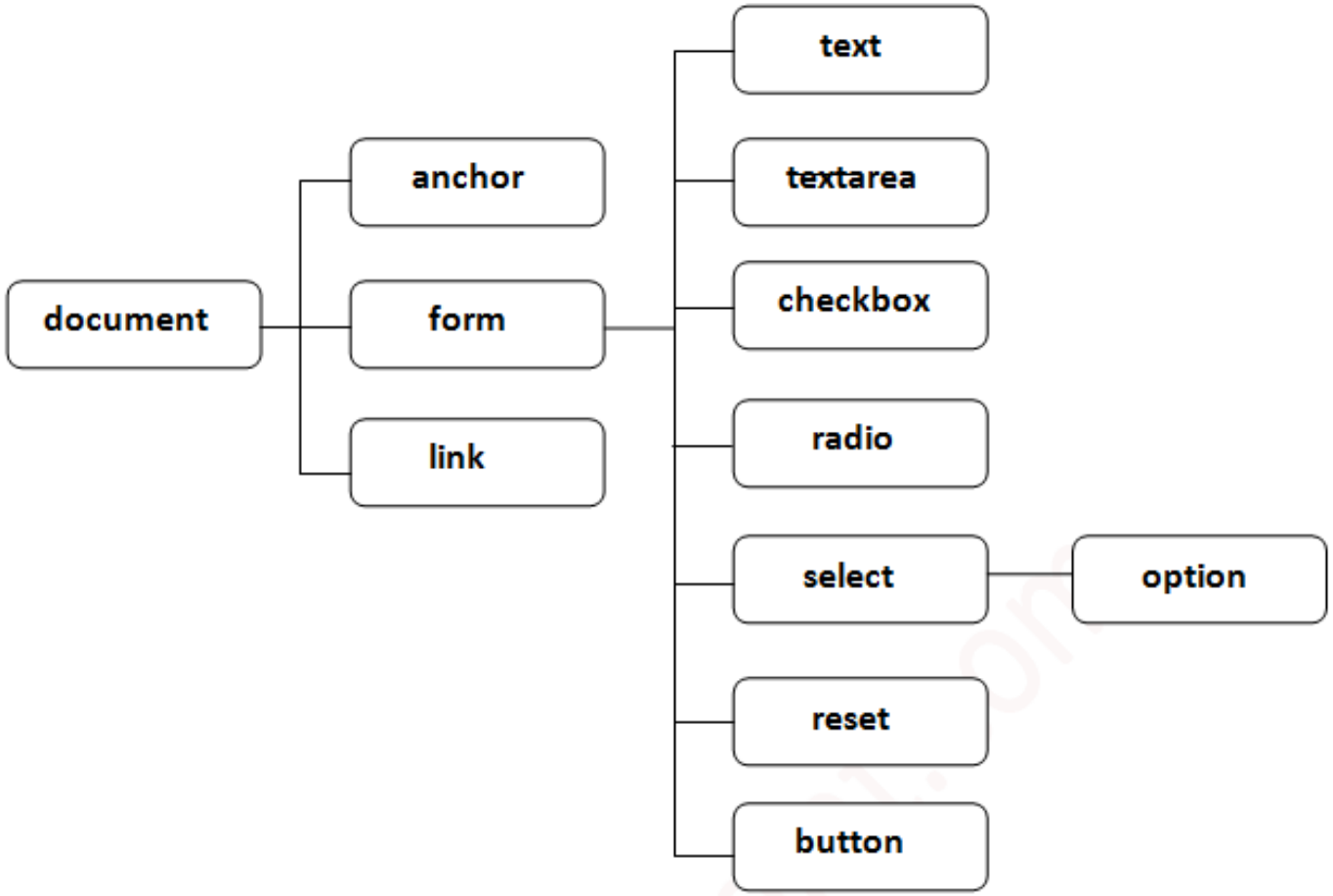


Document Object Model (DOM)

- The document object represents the whole html document.
- When html document is loaded in the browser, it becomes a document object. It is the root element that represents the html document.
- By the help of document object, we can add dynamic content to our web page.
- As mentioned earlier, it is the object of window. So

window.document is same as document

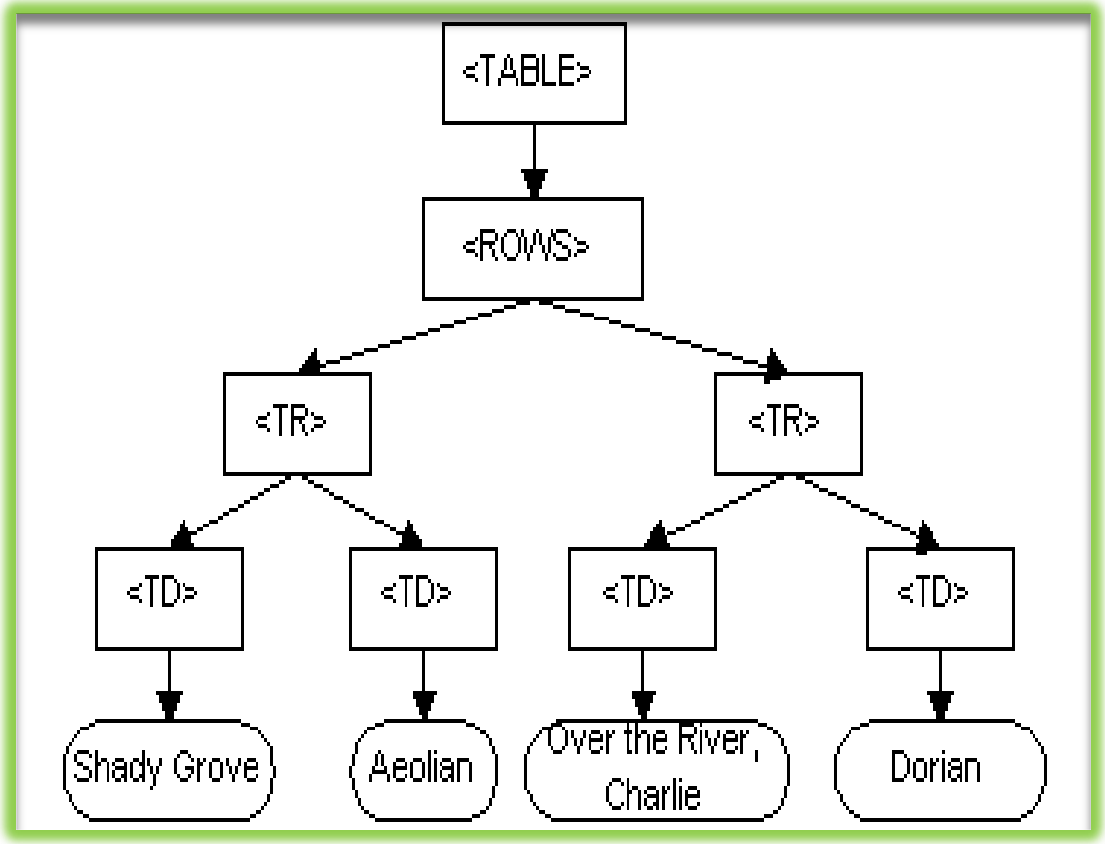
Document Object Model (DOM)



Properties of Document Object

- The Document Object Model is a programming API for documents. The object model itself closely resembles the structure of the documents it models.
- For instance, consider this table, taken from an HTML document:

```
<TABLE>
<ROWS>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</ROWS>
</TABLE>
```



Methods of document object

Method	Description
write("string")	writes the given string on the document.
writeln("string")	writes the given string on the document with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name

Accessing field value by document object

- In this example, we are going to get the value of input text by user. Here, we are using **document.form1.name.value** to get the value of name field.
- Here, **document** is the root element that represents the html document.
- **form1** is the name of the form.
- **name** is the attribute name of the input text.
- **value** is the property, that returns the value of the input text.

```
<script type="text/javascript">
function printvalue()
{
var name=document.form1.name.value;
alert("Welcome: "+name);
}
</script>
<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

innerText property of document

- The innerText property can be used to write the dynamic text on the html document. Here, text will not be interpreted as html text but a normal text.
- It is used mostly in the web pages to generate the dynamic content such as writing the validation message, password strength etc.
- For example, to display the password strength when releases the key after press.

innerText property of document-Example

```
<script type="text/javascript" >
function validate()
{    var msg;
     if(document.myForm.userPass.value.length>5)
     {    msg="good";    }
    else
    {    msg="poor";    }
    document.getElementById('mylocation').innerText=msg;
 }
  </script>
<form name="myForm">
<input type="password" value="" name="userPass" onkeyup="validate()">
Strength:<span id="mylocation">no strength</span>
</form>
```

Window Object

- Window object is a top-level object in Client-Side JavaScript.
- Window object represents the browser's window.
- It represents an open window in a browser.
- It supports all browsers.
- The document object is a property of the window object.
- So, typing window.document.write is same as document.write.
- All global variables are properties and functions are methods of the window object.

Window Object Properties

Property	Description
Document	It returns the document object for the window (DOM).
Frames	It returns an array of all the frames including iframes in the current window.
Closed	It returns the Boolean value indicating whether a window has been closed or not.
History	It returns the history object for the window.
innerHeight	It sets or returns the inner height of a window's content area.
innerWidth	It sets or returns the inner width of a window's content area.
Length	It returns the number of frames in a window.
Location	It returns the location object for the window.
Name	It sets or returns the name of a window.
Navigator	It returns the navigator object for the window.

Window Object Properties

Property	Description
Opener	It returns a reference to the window that created the window.
outerHeight	It sets or returns the outer height of a window, including toolbars/scrollbars.
outerWidth	It sets or returns the outer width of a window, including toolbars/scrollbars.
Parent	It returns the parent window of the current window.
Screen	It returns the screen object for the window.
screenX	It returns the X coordinate of the window relative to the screen.
screenY	It returns the Y coordinate of the window relative to the screen.
Self	It returns the current window.
Status	It sets the text in the status bar of a window.
Top	It returns the topmost browser window that contains frames.

Window Object Methods

Method	Description
alert()	It displays an alert box.
confirm()	It displays a dialog box.
prompt()	It displays a dialog box that prompts the visitor for input.
setInterval()	It calls a function or evaluates an expression at specified intervals.
setTimeout()	It evaluates an expression after a specified number of milliseconds.
clearInterval()	It clears a timer specified by setInterval().
clearTimeOut()	It clears a timer specified by setTimeout().
close()	It closes the current window.
open()	It opens a new browser window.
createPopup()	It creates a pop-up window.

Window Object Methods

Method	Description
focus()	It sets focus to the current window.
blur()	It removes focus from the current window.
moveBy()	It moves a window relative to its current position.
moveTo()	It moves a window to the specified position.
resizeBy()	It resizes the window by the specified pixels.
resizeTo()	It resizes the window to the specified width and height.
print()	It prints the content of the current window.
scrollBy()	It scrolls the content by the specified number of pixels.
scrollTo()	It scrolls the content to the specified coordinates.
focus()	It sets focus to the current window.

Opening and Closing Window

- The **window** object is supported by all browsers.
- It represents the browser's window.
-
- To open the window, javascript provides **open()** method.
- Syntax: window.open();
-
- To close the window, Javascript provides **close()** method.
- Syntax: window.close();

Opening Window - Example

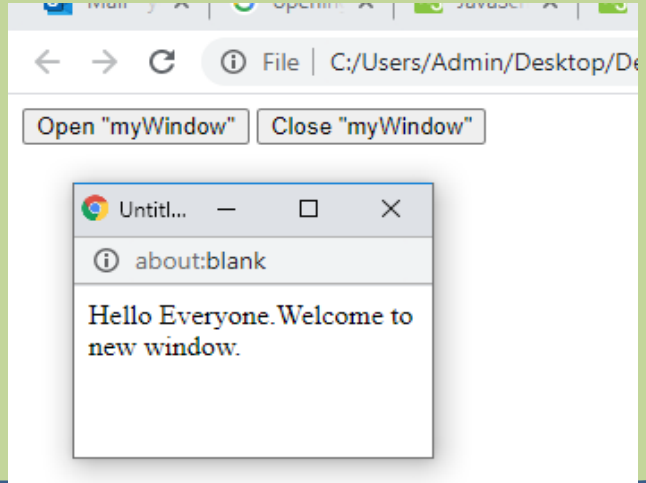
```
//save as hello.html

<html>
<body>
<script>
document.write("Hello Everyone!!!!");
</script>
</body>
</html>
```

```
//save as sample.html

<html>  <body>

<script>

var ob =window.open ("hello.html", "windowName",
"top=200,left=100,width=250,height=100,status");

</script>

</body>  </html>
```

Opening and Closing Window - Example

Code: Open a new window and close that window on button click event using open( ) and close ( ).

```
<html>    <body>
<button onclick="openWin()">Open "myWindow"</button>
<button onclick="closeWin()">Close "myWindow"</button>
<script>
var myWindow;
function openWin()
{
  myWindow = window.open("", "myWindow", "width=200,height=100");
  myWindow.document.write("<p>Hello Everyone.Welcome to new window.</p>");
}

function closeWin()
{
  myWindow.close();
}
</script>
</body>  </html>
```



Opening and Closing Window - Example

Code: to open vpt.edu.in website.

```
<html>
<body>
<button onclick="myFunction()">Open Windows</button>
<script>
function myFunction()
{
  window.open("http://www.vpt.edu.in/");
 }
</script>
</body>
</html>
```

Window Position

- A new window always displayed on the screen at the location which is specified by user and location is specified by setting the left and top properties of new window as shown below:

```
window.open("http://vpt.edu.in", "windowName", top=500,left=500,width=400,height=400");
```

- The **innerWidth** property returns the width of a window's content area.
- The **innerHeight** property returns the height of a window's content area.
- Syntax:
- window.innerWidth
  window.innerHeight

Window Position

- The **outerWidth** property returns the outer width of the browser window, including all interface elements (like toolbars/scrollbars).

- The **outerHeight** property returns the outer height of the browser window, including all interface elements (like toolbars/scrollbars).

- Syntax:

- window.outerWidth
window.outerHeight

- **These properties are read-only.**

Window Position- Example

Code: To retrieve the dimensions of window:

```
<html>
<body>
<div id="demo"></div>
<script>
var txt = "";
txt += "<p>innerWidth: " + window.innerWidth + "</p>";
txt += "<p>innerHeight: " + window.innerHeight + "</p>";
txt += "<p>outerWidth: " + window.outerWidth + "</p>";
txt += "<p>outerHeight: " + window.outerHeight + "</p>";
document.getElementById("demo").innerHTML = txt;
</script>
</body>
</html>
```

innerWidth: 606

innerHeight: 448

outerWidth: 1366

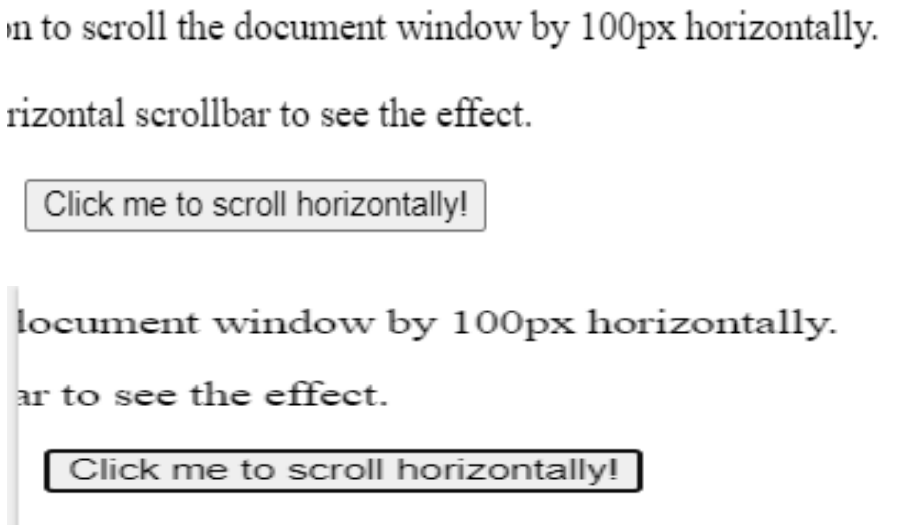
outerHeight: 728

Window Position

- Window provides two methods which also deal with positioning of windows named scrollBy() and moveTo() method.
- **scrollBy():** The scrollBy() method scrolls the document by the specified number of pixels.
- Syntax: window.scrollBy(xnum, ynum)
- The **moveTo()** method moves a window's left and top edge to the specified coordinates.
- Syntax: window.moveTo(*x, y*)
- The **focus()** method is used to give focus to an element (if it can be focused).
- Syntax: *HTMLElementObject*.focus()

Window Position: scrollBy() example

```
<html>  <head>
<style>
body
{
  width: 5000px;
}
button
{
  position: fixed;
}
</style>    </head>
<body>
<p>Click the button to scroll the document window by 100px horizontally.</p>
<p>Look at the horizontal scrollbar to see the effect.</p>
<button onclick="scrollWin()">Click me to scroll horizontally!</button><br><br>
<script>
function scrollWin()
{
  window.scrollBy(100, 0);
}
</script>  </body>   </html>
```



Window Position: moveTo() example

```
<html>  <body>

<p>Open "myWindow" and move the new window to the top left corner of the screen:</p>

<button onclick="openWin()">Open "myWindow"</button>
<button onclick="moveWin()">Move "myWindow"</button>

<script>
var myWindow;

function openWin()
{
  myWindow=window.open("", "myWindow", "width=200, height=100");
  myWindow.document.write("<p>This is 'myWindow'</p>");
}

function moveWin()
 {
  myWindow.moveTo(500, 100);
  myWindow.focus();
}
</script>    </body>    </html>
```

Learning Outcome 3 :
Student should understand properties and methods of window object.

Window.screen: contains information about the user's screen.

Proprties	Description
screen. availTop	Returns the top side of the area on the screen that is available for application windows.
screen.availLeft	Returns the left side of the area on the screen that is available for application windows.
screen.availWidth	returns the width of the visitor's screen, in pixels, minus interface features like the Windows Taskbar.
screen.availHeight	returns the height of the visitor's screen, in pixels, minus interface features like the Windows Taskbar.
screen.height	Returns the vertical resolution of the display screen in pixels.
screen.width	Returns the horizontal resolution of the display screen in pixels.
screen.left	Retrieves the horizontal offset of top-left corner of the current screen relative to the top-left corner of the main screen in pixels.
screen.top	Retrieves the vertical offset of top-left corner of the current screen relative to the top-left corner of the main screen in pixels.

Changing the contents of window

- We can change the content of opened new window as and when require.
- As a new window is created and opened using open() method of window object.
- In following example, we are creating only one object of window and each time same window remain open and content of widow changes.

```
<script>
function openWin1(ad)
{
  myWindow = window.open(ad, "myWindow", "width=500,height=500");
}
</script>
<button value="Google"
onclick="openWin1('https://www.google.com')">Google</button>
<button value="Vidyalankar"
onclick="openWin1('http://vpt.edu.in')">Vidyalanakr</button>
```

Scrolling a Web Page

- The scrollTo() method scrolls the document to the specified coordinates.
- Syntax: window.scrollTo(*xpos,ypos*);

Where,

Xpos: Required. The coordinate to scroll to, along the x-axis (horizontal), in pixels

Ypos: Required. The coordinate to scroll to, along the y-axis (vertical), in pixels

Scrolling a Web Page-Example

```
<html>  <head>
<style>
body
{
  width: 5000px;
  height: 5000px;
}
</style>   </head>
<body>
 <script>
function scrollHorizontal()
{
  window.scrollTo(100, 0);
}
function scrollVertical()
{
  window.scrollTo(0,100);
}
</script>
<p>Click the button to scroll the document window to 100 pixels horizontally and vertically.</p>

<button onclick="scrollHorizontal()">Click me to scroll horizontally !</button>
<br><br>
<button onclick="scrollVertical()">Click me to scroll vertically!</button>
 </body>
</html>
```

Opening multiple windows at once

```
<html>  <head>
<title>window </title>
<script type="text/javascript">
function show()
{
for( i=0; i< 250 ; i=i+50)
{
var x=i+50;
var y=i+50;
winobj=window.open('','win' +i, 'top='+x+ ',left='+y+',width=300,height=200');
}
}
</script>
</head>
<body>
<input type="multiple" value="Show Multiple Windows" type="button" onclick="show()"/>
</body>  </html>
```

Creating a web page in new window

```
<html>  <head>
<title>window </title>    <script type="text/javascript">
function newWindow()
{
winobj=window.open("","winname","width=300, height=300, left=200, top=200");
winobj.document.write('<html>');
winobj.document.write('<head>');
winobj.document.write('<title> writing Content</title>');
winobj.document.write('</head>');
winobj.document.write('<body>');
winobj.document.write('<form action="" method="post">');
winobj.document.write('<p>');
winobj.document.write('Enter Inst Name:<input type="text" name="iname">');
winobj.document.write('<br>');
winobj.document.write('<input type="submit" name="submit">');
winobj.document.write('</p>');
winobj.document.write('</form>');
winobj.document.write('</body>');
winobj.document.write('</html>');
winobj.focus();
}
</script>  </head>
<body>  <input type="button" value="New value" onclick="newWindow()">
</body>   </html>
```

Javascript in URL

- Another way that JavaScript code can be included on the client side is in a URL following the javascript: pseudo-protocol specifier. This special protocol type specifies that the body of the URL is arbitrary JavaScript code to be interpreted by the JavaScript interpreter. If the JavaScript code in a javascript: URL contains multiple statements, the statements must be separated from one another by semicolons. Such a URL might look like the following:

```
javascript:var now = new Date(); " The time is:" + now;
```

- When the browser "loads" one of these JavaScript URLs, it executes the JavaScript code contained in the URL and displays the "document" referred to by the URL. This "document" is the string value of the last JavaScript statement in the URL. This string will be formatted and displayed just like any other document loaded into the browser. More commonly, a JavaScript URL will contain JavaScript statements that perform actions but return no value.

- For example:

```
javascript:alert("Hello World!");
```

Timers

- The window object allows execution of code at specified time intervals.
- These time intervals are called timing events.
- The two key methods to use with JavaScript are:
  - setTimeout(function, milliseconds)
    Executes a function, after waiting a specified number of milliseconds.
  - setInterval(function, milliseconds)
    Same as setTimeout(), but repeats the execution of the function continuously.
- The setTimeout() and setInterval() are both methods of the HTML DOM Window object.

Timers-Example

Code: Click a button. Wait 3 seconds, and the page will alert "Hello":

```
<html>
<body>
<p>Click "Try it". Wait 3 seconds, and the page will alert "Hello".</p>
<button onclick="setTimeout(myFunction, 3000);">Try it</button>
<script>
function myFunction()
{
  alert('Hello');
}
</script>
</body>
</html>
```

How to Stop the Execution?

- The clearTimeout() method stops the execution of the function specified in setTimeout().

  window.clearTimeout(*timeoutVariable*)

- The window.clearTimeout() method can be written without the window prefix.
- The clearTimeout() method uses the variable returned from setTimeout():

```
myVar = setTimeout(function, milliseconds);
clearTimeout(myVar);
```

- If the function has not already been executed, you can stop the execution by calling the clearTimeout() method:

The setInterval() Method

- The setInterval() method repeats a given function at every given time-interval.

```
window.setInterval(function, milliseconds);
```

- The window.setInterval() method can be written without the window prefix.
- The first parameter is the function to be executed.
- The second parameter indicates the length of the time-interval between each execution.
- This example executes a function called "myTimer" once every second (like a digital watch).

The clearInterval() method

- The clearInterval() method stops the executions of the function specified in the setInterval() method.

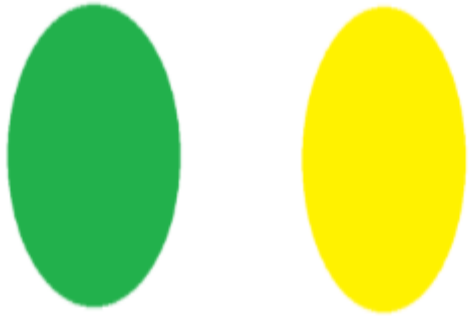
window.clearInterval(*timerVariable*)

- The window.clearInterval() method can be written without the window prefix.
- The clearInterval() method uses the variable returned from setInterval():

```
myVar = setInterval(function, milliseconds);
clearInterval(myVar);
```

Code: Rotating images using setTimeout(0 method. Following example is using an array of images. Using setTimeout(0 method images will be rotated after 1 second.

```
<html>  <body>  <script>
var i,imgs,pic;
function init()
{
pic=document.getElementById("pic");
imgs=["red.png","blue.png","green.png","yellow.png"];
i=0;
rotate();
}
 function rotate()
{
pic.src=imgs[i];
(i==(imgs.length-1))?i=0:i++;
setTimeout(rotate,1000);
}
document.addEventListener("DOMContentLoaded",init,false);
</script>  </head>  <body>
<img id="pic" width="300" height="300">  </body>  </html>
```



Code: Moving car using setTimeout( ) and clearTimeout( ) methods.

```
<html>   <head>
<title>Animation </title>
<script type="text/javascript">
var obj=null;
var animate;
function init()
{
obj=document.getElementById('car');
obj.style.position='relative';
obj.style.left='0px';
}
 function start()
{
obj.style.left=parseInt(obj.style.left)+    10    +
'px';
animate=setTimeout(start,20);
}
function stop()
{
clearTimeout(animate);
obj.style.left='0 px';
}
window.onload=init;
</script>
</head>
<body>
<img id="car" src="car.jpg">
<br><br>
<input value="Start" type="button"
onclick="start()"/>
<input value="Stop" type="button"
onclick="stop()"/>
</body>
</html>
```