**LAB – 7**

**JavaScript HTML DOM:**

With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

**The HTML DOM (Document Object Model):**

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.The **HTML DOM** model is constructed as a tree of **Objects**:

### The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

* JavaScript can change all the HTML elements in the page
* JavaScript can change all the HTML attributes in the page
* JavaScript can change all the CSS styles in the page
* JavaScript can remove existing HTML elements and attributes
* JavaScript can add new HTML elements and attributes
* JavaScript can react to all existing HTML events in the page
* JavaScript can create new HTML events in the page

# JavaScript HTML DOM - Changing CSS

The HTML DOM allows JavaScript to change the style of HTML elements.

**Changing HTML Style**

To change the style of an HTML element, use this syntax:

document.getElementById(*id*).style.*property*=*new style*

## Using Events

The HTML DOM allows you to execute code when an event occurs.

Events are generated by the browser when "things happen" to HTML elements:

* An element is clicked on
* The page has loaded
* Input fields are changed

**PROGRAM:**

<!DOCTYPE html>

<html>

<head><title>changing html styles using html events</title>

</head>

<body>

<h1 id="id2"><b>INTRODUCTION TO HTML DOM</b></h1>

<p id="id1">

The HTML DOM allows you to execute that event section of code when an event occurs.<br>

Events are generated by the browser when "alert send " to HTML elements:<br>

An element is clicked on the page has loaded.<br>

Input fields are changed<br></p>

<script>

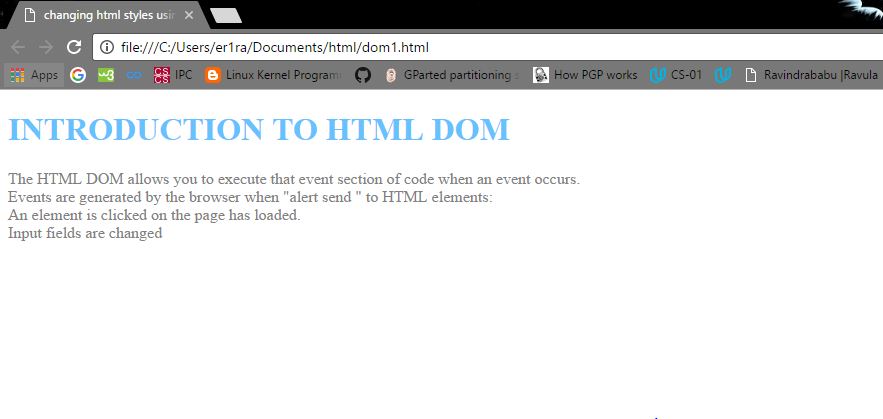
document.getElementById("id2").style.color = "#66c2ff";

document.getElementById("id1").style.color = "grey";

</script>

</body>

</html>



**JavaScript Window - The Browser Object Mode:**

The Browser Object Model (BOM) allows JavaScript to "talk to" the browser.

**The Browser Object Model (BOM)**

There are no official standards for the **B**rowser **O**bject **M**odel (BOM).Since modern browsers have implemented (almost) the same methods and properties for JavaScript interactivity, it is often referred to, as methods and properties of the BOM.

**The Window Object:**

The **window** object is supported by all browsers. It represents the browser's window.All global JavaScript objects, functions, and variables automatically become members of the window object.Global variables are properties of the window object.Global functions are methods of the window object.Even the document object (of the HTML DOM) is a property of the window object:

window.document.getElementById("header");

## Other Window Methods:

Some other methods:

* window.open() - open a new window
* window.close() - close the current window
* window.moveTo() -move the current window
* window.resizeTo() -resize the current window

**PROGRAM:**

<!DOCTYPE html>

<html>

<head><title>use of Javascript window objects</title>

</head>

<body>

<p>Click the button to open a new window called "Message\_Window" with some text.</p>

<button onclick="open\_Win()">Open "my\_Window"</button>

<button onclick="close\_Win()">Close "my\_Window"</button>

<script>

varmy\_Window;

function open\_Win()

{

my\_Window = window.open("", "my\_Window", "width=200,height=100");

my\_Window.document.write("<p>This is 'mybrowser'</p>");

my\_Window.opener.document.write("<p>This is the source browser!</p>");

}

function close\_Win()

{

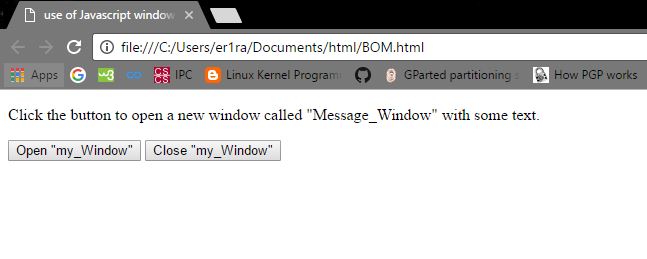
my\_Window.close();

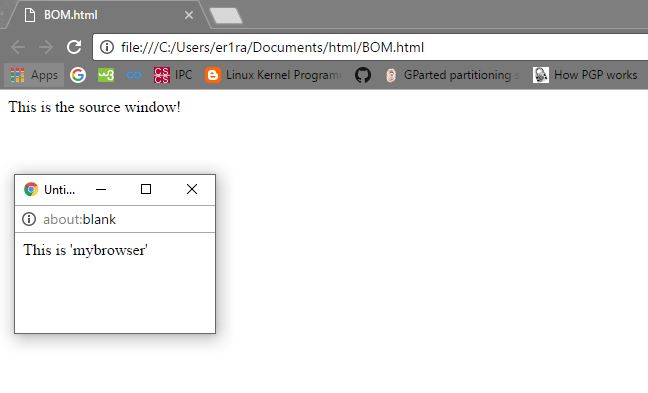
}

</script>

</body>

</html>





# JavaScript Window Location

The window.location object can be used to get the current page address (URL) and to redirect the browser to a new page.

## Window Location

The **window.location** object can be written without the window prefix.

Some examples:

* window.location.href returns the href (URL) of the current page
* window.location.hostname returns the domain name of the web host
* window.location.pathname returns the path and filename of the current page
* window.location.protocol returns the web protocol used (http: or https:)
* window.location.assign loads a new document

**PROGRAM:**

<!DOCTYPE html>

<html>

<body>

<p>Display the entire standard URL of the current page.</p>

<p id="Demo"></p>

<p>Display the name of path of the current URL.</p>

<p id="id1"></p>

<script>

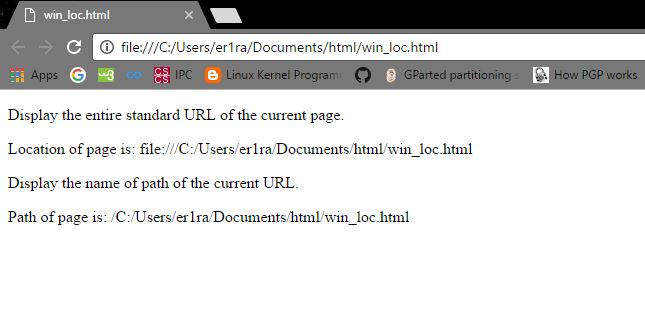
document.getElementById("Demo").innerHTML = "Location of page is: " + window.location.href;

document.getElementById("id1").innerHTML = "Path of page is: " + window.location.pathname;

</script>

</body>

</html>



# JavaScript Window Navigator

The window.navigator object contains information about the visitor's browser

## Window Navigator

The **window.navigator** object can be written without the window prefix.

Some examples:

* navigator.appName
* navigator.appCodeName
* navigator.platform

**PROGRAM:**

<!DOCTYPE html>

<html>

<body>

<h1>The Navigator Object</h1>

<p>The cookie Enabled property returns true if cookies are enabled:</p>

<p id="Demo"></p>

<p>The appVersion property returns information about the browser version:</p>

<p id="id1"></p>

<script>

document.getElementById("Demo").innerHTML ="navigator.cookie Enabled is " + navigator.cookieEnabled;

document.getElementById("id1").innerHTML = navigator.appVersion;

</script>

</body>

</html>



# JavaScript Window History

The window.history object contains the browsers history.

**Window History**

The **window.history** object can be written without the window prefix.

To protect the privacy of the users, there are limitations to how JavaScript can access this object.

Some methods:

* history.back() - same as clicking back in the browser
* history.forward() - same as clicking forward in the browser

# JavaScript Cookies:

Cookies are data, stored in small text files, on your computer.When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.Cookies were invented to solve the problem "how to remember information about the user":

* When a user visits a web page, his name can be stored in a cookie.
* Next time the user visits the page, the cookie "remembers" his name.

**PROGRAM:**

<html>

<head>

<script type>

function WriteCookie()

{

if( document.myform.customer.value == "" )

{

alert("Enter some value!");

return;

}

cookievalue= escape(document.myform.customer.value) + ";";

document.cookie="name=" + cookievalue;

document.write ("Setting Cookies : " + "name=" + cookievalue );

}

</script>

</head>

<body>

<form name="myform" action="">

Enter name: <input type="text" name="customer"/>

<input type="button" value="Set Cookie" onclick="WriteCookie();"/>

</form>

</body>

</html>

