**Workshop**

**What is HTML?**

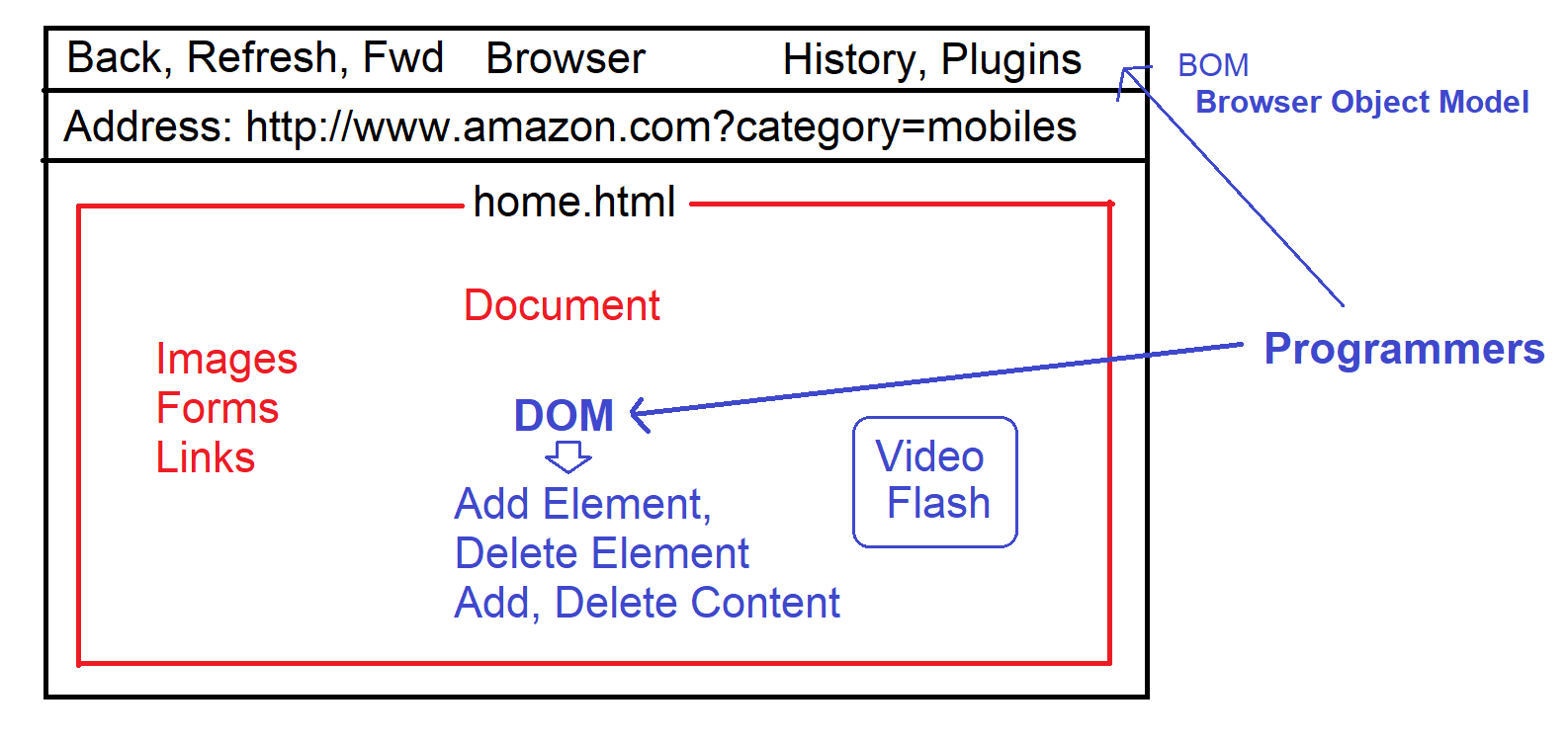
* HTML is a presentation language.
* Hyper Text Markup Language.
* Term Hyper is derived from Greek Term, which means “Beyond”
* Hyper Text is the text that comprises of content beyond what you see.
* Markup means marking up our information for presentation.
* HTML comprises of elements arranged in a hierarchy called DOM.   
  (Document Object Model)

**What is the Document Object Model [DOM]?**

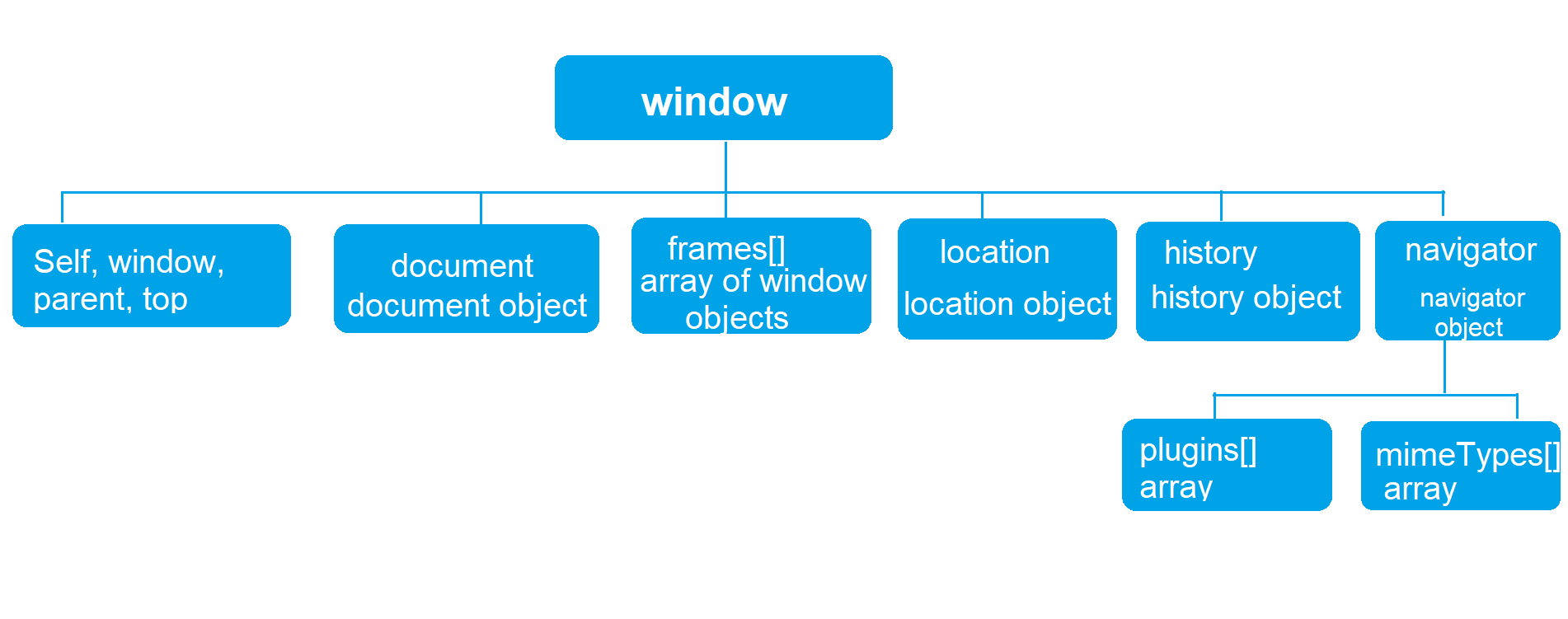
* The Document Object Model (DOM) is a programming API for HTML and XML documents.
* It defines the logical structure of documents.
* It specifies a way for accessing and manipulating HTML and XML document.
* HTML presents all its elements in a hierarchy called DOM.
* With DOM programmers can create and build documents, navigate their structure, and add, modify or delete elements and content.
* HTML presents DOM and JavaScript Manipulates DOM
* HTML DOM is static and JavaScript makes the static DOM into Dynamic DOM.

**What is the Browser Object Model [BOM]?**

* It represents the browser model.
* It defines a set of objects that are used to manipulate the browser.
* Programmer uses BOM to handle interaction with regard to browser.
* Programmer can know about browser and its capabilities by using BOM.
* BOM is programming API for Browser manipulations.



**Browser Object Model**



|  |  |
| --- | --- |
| **Browser Object** | **Description** |
| window | It provides a set of properties and methods which are used to manipulate the browser window, like open window, close, minimize window etc. |
| location | It provides a set of properties and methods which are used to know client location details, like URL, IP address, Path, Server Name, port number etc. |
| history | It provides a set of properties and method which are used to manipulate the browser history. |
| navigator | It provides a set of properties and methods which are used to know the client browser details, like browser family name, version, plugins, MIME types, cookies etc. |
| document | It provides access to the DOM model. It provides set of properties and methods that are used to manipulate the HTML document in browser. |

**Setup Environment**

* Downloading and Install any Editor
  + Visual Studio Code
  + Web Strom
  + Sublime
  + Eclipse etc.

<https://code.visualstudio.com/Download>

* Install any Live Server Plugin into your editor or you can also view in a browser.
* Visual Studio Code Editor
  + Install Extension by name “Live Server”
* **Create a new Folder for our workshop and Open in your editor**

**Browser Object Model**

* window
* location
* navigator
* history
* document

**window object**

* It provides a set of properties and methods that are used to manipulate the browser window.

|  |  |
| --- | --- |
| **Member** | **Description** |
| open() | It opens any document in a new window.  Syntax:  window.open(“path”, “title”, “options”) |
| close() | It closes the current window or any specific window in workspace. |
| print() | It opens printer properties to print the page. |

Ex:

<!DOCTYPE html>

<html>

<head>

<title>BOM</title>

<script>

function OpenImage(){

window.open("Images/speaker.jpg","JBL Speaker","width=600 height=400");

}

function CloseWindow(){

window.close();

}

function PrintWindow(){

window.print();

}

function OpenShoe(){

window.open("Images/shoe.jpg","Nike","width=500 height=400");

}

</script>

<style>

img:hover {

cursor:grab;

}

</style>

</head>

<body>

<h2>Window Object</h2>

<button onclick="OpenImage()" >Open</button>

<button onclick="CloseWindow()">Close</button>

<button onclick="PrintWindow()">Print</button>

<div onclick="OpenShoe()">

<img src="Images/shoe.jpg" width="100" height="100">

</div>

</body>

</html>

**location object**

* It provides a set of properties and methods that are used to get or set the client location.

|  |  |
| --- | --- |
| **Member** | **Description** |
| host | Gets the server name or IP address |
| port | Gets the port number |
| protocol | Gets the current protocol |
| href | Gets the current URL |
| pathname | Gets the current file path. |
| reload() | It reloads the current location [Refresh the page] |
| search | It gets the query string. |

Ex:

<!DOCTYPE html>

<html>

<head>

<script>

function GetDetails()

{

document.getElementById("host").innerText = location.host;

var msg;

if(location.protocol=="http:") {

msg = "You are using Live Web Server";

} else {

msg = "You are using File Server";

}

document.getElementById("protocol").innerHTML = location.protocol + "<br>" + msg;

document.getElementById("url").innerText = location.href;

}

function GotoPage() {

location.href = "home.html";

}

</script>

<style>

dt {

background-color:darkcyan;

color:white;

margin-top: 20px;

}

</style>

</head>

<body>

<h2>Location Details</h2>

<button onclick="GetDetails()">Get Details</button>

<button onclick="GotoPage()">View Window Example</button>

<dl>

<dt>Server Name / IP Address</dt>

<dd id="host"></dd>

<dt>Protocol</dt>

<dd id="protocol"></dd>

<dt>URL</dt>

<dd id="url"></dd>

</dl>

</body>

</html>

**Navigator Object**

* It provides a set of properties and methods that are used to know the browser details.

|  |  |
| --- | --- |
| appName | Returns the browser family name. [Netscape] |
| platform | Get the details of supported platforms. |
| cookieEnabled | Returns true if cookies are enabled. |
| plugins[] | Returns the list of plugins installed in browser. |
| mimeTypes[] | Returns the list supported file types in browser. |

Ex:

<!DOCTYPE html>

<html>

<head>

<title>Navigator</title>

<script>

function Verify(){

var msg;

if(navigator.cookieEnabled)

{

msg = "Cookies Enabled.. You can Continue - You are using " + navigator.language;

} else {

msg = "Your Browser Disabled Cookies - Please Enable";

}

document.getElementById("msg").innerHTML = msg;

}

</script>

</head>

<body>

<button onclick="Verify()">Verify Cookies</button>

<h3 id="msg">

</h3>

</body>

</html>

Ex: To Get All Plugins

<!DOCTYPE html>

<html>

<head>

<script>

function f1(){

for(var i=0; i<navigator.plugins.length; i++)

{

document.write(navigator.plugins[i].name + "<br>");

}

}

f1();

</script>

</head>

<body>

</body>

</html>

Ex: To Verify any specific plugin

<!DOCTYPE html>

<html>

<head>

<script>

function f1(){

if(navigator.plugins['Chrome PDF Viewer']==undefined)

{

alert("You don't have PDF Plugin Please Install");

location.href="http://www.adobe.com/download";

} else {

document.write("You can view PDF Documents");

}

}

f1();

</script>

</head>

<body>

</body>

</html>

**History Object**

* It provides set of properties and method that are used to access browser history and navigate to previous and next pages in history.

|  |  |
| --- | --- |
| **Member** | **Description** |
| length | Gets the total count of pages in browser history. |
| back() | Moves to previous page in browser history. |
| forward() | To next page in browser history. |
| goto() | To move to any specific page in browser history. |

Ex:

<!DOCTYPE html>

<html>

<head>

<script>

function f1(){

if(history.length>3) {

alert("You can view Max 3 Pages- Please Register for More");

location.href="register.html";

} else {

document.write("You can view max 3 page only..");

}

}

f1();

</script>

</head>

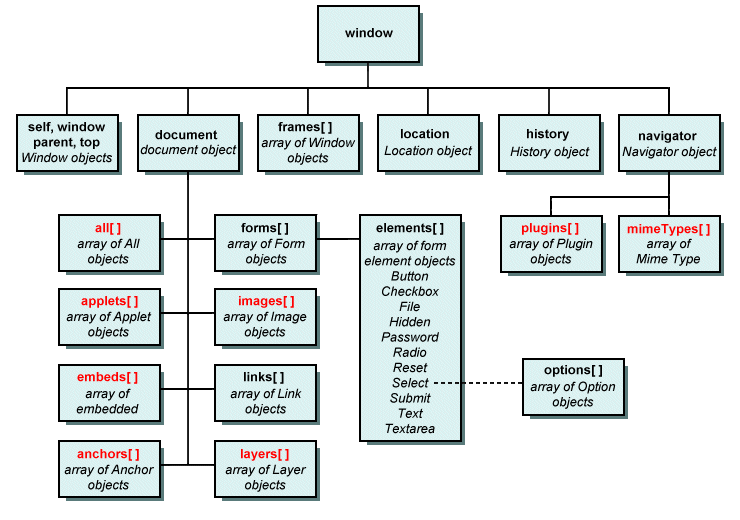
<body>

</body>

</html>

**Document Object**

* It provides the set of properties and method that are used to manipulate the HTML document.
* Adding elements / Data
* Removing elements / Data
* Modifying elements/ Data etc.
* This document exposes the DOM [Document Object Model]
* Document object provides options used by developer to manipulate HTML document.
* DOM is a hierarchy of elements in document. [Root and Child Nodes]
* HTML elements that you use for presentation are defined as Nodes under DOM.



|  |  |
| --- | --- |
| **Node** | **Description** |
| Element Node | An element, as it exists in DOM. |
| Root Node | The top node in the tree. In the case of HTML, it is “<html>” |
| Child Node | A node directly inside another node. For example “<img>” is a child of “<figure>” |
| Descendant Node | A node anywhere inside another node.  <body>  <figure>  <img>  </figure>  </body>   * **<img> is child node of <figure>** * **<img> is descendant of <body>** * **<figure> is child node of <body>** |
| Parent Node | A node which has another node inside.  <body> is parent node. |
| Sibling Nodes | Nodes that sit on the same level in the DOM tree.  <body>  <img>  <p>  </body>   * <img> is sibling of <p> |
| Text Node | A node containing a text string. |

* Programming can use JavaScript to manipulate DOM.
* JavaScript allows the developer to access the nodes by using the DOM hierarchy.

Ex:

<!DOCTYPE html>

<html>

<head>

<script>

function bodyload()

{

window.document.images[0].src="Images/shoe.jpg";

window.document.forms[1].elements[1].value="Login";

window.document.forms[1].elements[0].value= "john@gmail.com";

window.document.forms[0].elements[1].value="Register";

}

</script>

</head>

<body onload="bodyload()">

<img width="100" height="100" border="1">

<div>

<form>

<h2>Register</h2>

<input type="text">

<input type="button">

</form>

</div>

<div>

<form>

<h2>Login</h2>

<input type="email">

<input type="button">

</form>

</div>

</body>

</html>

* When ever the position of any element changes in page. You have to update its index in the logic.
* To overcome the issue, you can use document methods like
  + document.getElementById()
  + document.getElementByName()
  + document.getElementByTagName()
  + document.getElementByClassName() etc.
* You can also used reference name for elements. You have to configure the complete parent and child node hierarchy in order access any child element.

**Ex: Referring Name**

<!DOCTYPE html>

<html>

<head>

<script>

function bodyload()

{

pic.src="Images/speaker.jpg";

frmRegister.btnRegister.value="Register";

frmLogin.btnLogin.value="Login";

}

</script>

</head>

<body onload="bodyload()">

<img name="pic" width="100" height="100" border="1">

<div>

<form name="frmRegister">

<h2>Register</h2>

<input name="txtName" type="text">

<input name="btnRegister" type="button">

</form>

</div>

<div>

<form name="frmLogin">

<h2>Login</h2>

<input name="txtEmail" type="email">

<input name="btnLogin" type="button">

</form>

</div>

</body>

</html>

Ex: **Refer by ID**

<!DOCTYPE html>

<html>

<head>

<script>

function bodyload()

{

document.getElementById("pic").src="Images/shoe.jpg";

document.getElementById("btnRegister").value="Register";

document.getElementById("btnLogin").value="Login";

}

</script>

</head>

<body onload="bodyload()">

<img name="pic" id="pic" width="100" height="100" border="1">

<div>

<form id="frmRegiter" name="frmRegister">

<h2>Register</h2>

<input id="txtName" name="txtName" type="text">

<input id="btnRegister" name="btnRegister" type="button">

</form>

</div>

<div>

<form id="frmLogin" name="frmLogin">

<h2>Login</h2>

<input name="txtEmail" type="email">

<input id="btnLogin" name="btnLogin" type="button">

</form>

</div>

</body>

</html>

Ex: To all document properties

<script>

function f1(){

for(var property in document)

{

document.write(property + "<br>");

}

}

f1();

</script>

location  
implementation  
URL  
documentURI  
compatMode  
characterSet  
charset  
inputEncoding  
contentType  
doctype  
documentElement  
xmlEncoding  
xmlVersion  
xmlStandalone  
domain  
referrer  
cookie  
lastModified  
readyState  
title  
dir  
body  
head  
images  
embeds  
plugins  
links  
forms  
scripts  
currentScript  
defaultView  
designMode  
onreadystatechange  
anchors  
applets  
fgColor  
linkColor  
vlinkColor  
alinkColor  
bgColor  
all  
scrollingElement  
onpointerlockchange  
onpointerlockerror  
hidden  
visibilityState  
wasDiscarded  
featurePolicy  
webkitVisibilityState  
webkitHidden  
onbeforecopy  
onbeforecut  
onbeforepaste  
onfreeze  
onresume  
onsearch  
onsecuritypolicyviolation  
onvisibilitychange  
fonts  
oncopy  
oncut  
onpaste  
activeElement  
styleSheets  
pointerLockElement  
fullscreenElement  
adoptedStyleSheets  
onabort  
onblur  
oncancel  
oncanplay  
oncanplaythrough  
onchange  
onclick  
onclose  
oncontextmenu  
oncuechange  
ondblclick  
ondrag  
ondragend  
ondragenter  
ondragleave  
ondragover  
ondragstart  
ondrop  
ondurationchange  
onemptied  
onended  
onerror  
onfocus  
onformdata  
oninput  
oninvalid  
onkeydown  
onkeypress  
onkeyup  
onload  
onloadeddata  
onloadedmetadata  
onloadstart  
onmousedown  
onmouseenter  
onmouseleave  
onmousemove  
onmouseout  
onmouseover  
onmouseup  
onmousewheel  
onpause  
onplay  
onplaying  
onprogress  
onratechange  
onreset  
onresize  
onscroll  
onseeked  
onseeking  
onselect  
onstalled  
onsubmit  
onsuspend  
ontimeupdate  
ontoggle  
onvolumechange  
onwaiting  
onwebkitanimationend  
onwebkitanimationiteration  
onwebkitanimationstart  
onwebkittransitionend  
onwheel  
onauxclick  
ongotpointercapture  
onlostpointercapture  
onpointerdown  
onpointermove  
onpointerup  
onpointercancel  
onpointerover  
onpointerout  
onpointerenter  
onpointerleave  
onselectstart  
onselectionchange  
onanimationend  
onanimationiteration  
onanimationstart  
ontransitionend  
children  
firstElementChild  
lastElementChild  
childElementCount  
fullscreenEnabled  
fullscreen  
onfullscreenchange  
onfullscreenerror  
webkitIsFullScreen  
webkitCurrentFullScreenElement  
webkitFullscreenEnabled  
webkitFullscreenElement  
onwebkitfullscreenchange  
onwebkitfullscreenerror  
rootElement  
getElementsByTagName  
getElementsByTagNameNS  
getElementsByClassName  
createDocumentFragment  
createTextNode  
createCDATASection  
createComment  
createProcessingInstruction  
importNode  
adoptNode  
createAttribute  
createAttributeNS  
createEvent  
createRange  
createNodeIterator  
createTreeWalker  
getElementsByName  
open  
close  
write  
writeln  
hasFocus  
execCommand  
queryCommandEnabled  
queryCommandIndeterm  
queryCommandState  
queryCommandSupported  
queryCommandValue  
clear  
captureEvents  
releaseEvents  
exitPointerLock  
createElement  
createElementNS  
caretRangeFromPoint  
getSelection  
elementFromPoint  
elementsFromPoint  
getElementById  
prepend  
append  
querySelector  
querySelectorAll  
exitFullscreen  
webkitCancelFullScreen  
webkitExitFullscreen  
createExpression  
createNSResolver  
evaluate  
onpointerrawupdate  
timeline  
hasStorageAccess  
requestStorageAccess  
getAnimations  
pictureInPictureElement  
pictureInPictureEnabled  
exitPictureInPicture  
ELEMENT\_NODE  
ATTRIBUTE\_NODE  
TEXT\_NODE  
CDATA\_SECTION\_NODE  
ENTITY\_REFERENCE\_NODE  
ENTITY\_NODE  
PROCESSING\_INSTRUCTION\_NODE  
COMMENT\_NODE  
DOCUMENT\_NODE  
DOCUMENT\_TYPE\_NODE  
DOCUMENT\_FRAGMENT\_NODE  
NOTATION\_NODE  
DOCUMENT\_POSITION\_DISCONNECTED  
DOCUMENT\_POSITION\_PRECEDING  
DOCUMENT\_POSITION\_FOLLOWING  
DOCUMENT\_POSITION\_CONTAINS  
DOCUMENT\_POSITION\_CONTAINED\_BY  
DOCUMENT\_POSITION\_IMPLEMENTATION\_SPECIFIC  
nodeType  
nodeName  
baseURI  
isConnected  
ownerDocument  
parentNode  
parentElement  
childNodes  
firstChild  
lastChild  
previousSibling  
nextSibling  
nodeValue  
textContent  
hasChildNodes  
getRootNode  
normalize  
cloneNode  
isEqualNode  
isSameNode  
compareDocumentPosition  
contains  
lookupPrefix  
lookupNamespaceURI  
isDefaultNamespace  
insertBefore  
appendChild  
replaceChild  
removeChild  
addEventListener  
removeEventListener  
dispatchEvent

**DOM Manipulation**

* Access and use HTML elements.
* Configure and Modify their data and styles.
* Removing Elements.

Ex:

* In modern programming we use “querySelector()” to access HTML elements. As it can use CSS selectors.

<!DOCTYPE html>

<html>

<head>

<script>

function bodyload()

{

var link = document.querySelector("a");

link.textContent = "NareshIT";

link.href = "http://www.nareshit.in";

}

</script>

</head>

<body onload="bodyload()">

<a></a>

</body>

</html>

**Creating and placing new nodes:**

* Create element by using “document.createElement()”
* Add element as child by using “appendChild()”

<!DOCTYPE html>

<html>

<head>

<script>

function bodyload()

{

var sect = document.querySelector("section");

var para = document.createElement("p");

para.textContent = "Hello ! Welcome to NareshIT";

sect.appendChild(para);

}

</script>

</head>

<body onload="bodyload()">

<section>

</section>

</body>

</html>

Ex: Create element by using its Base class

<!DOCTYPE html>

<html>

<head>

<script>

function bodyload()

{

var sect = document.querySelector("section");

var para = document.createElement("p");

para.textContent = "Hello ! Welcome to NareshIT";

sect.appendChild(para);

var fig = document.querySelector("figure");

var pic = new Image();

pic.width="100";

pic.height="100";

pic.border="1";

pic.src="Images/speaker.jpg";

fig.appendChild(pic);

}

</script>

</head>

<body onload="bodyload()">

<figure>

</figure>

<section>

</section>

</body>

</html>