M T W T F S S

Page No.:

Date:

YOUVA

JavaScript

BOM

DOM

JS BOM

Page No.:
Date:
YOUVA

Bom - Browser object model.

It represents browser's window, All JS objects, function and variable automatically become member of **window object**

i) Global variable are properties of window object
ii) Global functions are methods of window object.

```
window.document.getElementById("header");
        Same as
document.getElementById("header");
```

JS Window Screen

window.screen can be written without window.

Screen.width - returns visitor screen in pixels

screen.height - returns visitors screen height in pixels

screen.availWidth - returns screen width minus features like taskbar etc (interface features).

Screen.availHeight - returns screen height minus interface features.

Screen.colorDepth - returns number of bits used to display one color.

screen.pixelDepth - returns pixel depth of the screen

JS Location

window.location has current web page address(URL) and to get redirect the browser to new page.

window.location can be written without window

1) location.href - return (URL) of current page
2) location.hostname - return domain name (www.google.com)
3) location.pathname - return path & file
4) location.protocol - return web protocol (http/https)
5) location.assign() - loads new document

```
<script>
    function newD(){
        window.location.assign("https://www.google.com");
    }
</script>

<body>
<input "button" "load new" newD()>
</body>
```

JS History

M T W T F S S
Page No.:
Date:
YOUVA

`window.history` has browsers history.

window.history can be written without window.

1) history.back() – Go Back.
2) history.forward() – Go forward.

```
<head>
<script>
    function Goforward(){
        window.history.forward();
    }
    function GoBack(){
        window.history.back();
    }
</script>
</head>

<body>
<input type="button" value="Forward" onclick="Goforward()">
<input type="button" value="Back" onclick="GoBack()">
</body>
```

Js Navigator

window.navigator can be used without window Prefix.

navigator.appName - return appname which is "Netscape"
navigator.appCodeName - return codeName which is "Mozilla"
navigator.platform - return OS
navigator.cookieEnabled - return boolean if cookie enable
navigator.product - return product name i.e "Gecko" os
navigator.appVersion - return version
navigator.userAgent - return user agent (whole info in for
navigator.language - return browser's language
navigator.onLine - return boolean if online
navigator.javaEnabled() - return if Java enabled

```
<p id = "demo"></p>
<p id = "demo2"></p>
<script>
    document.getElementById( ).innerHTML =
                navigator.appCodeName;

    document.getElementById( ).innerHTML =
                navigator.javaEnabled();
</script>
```

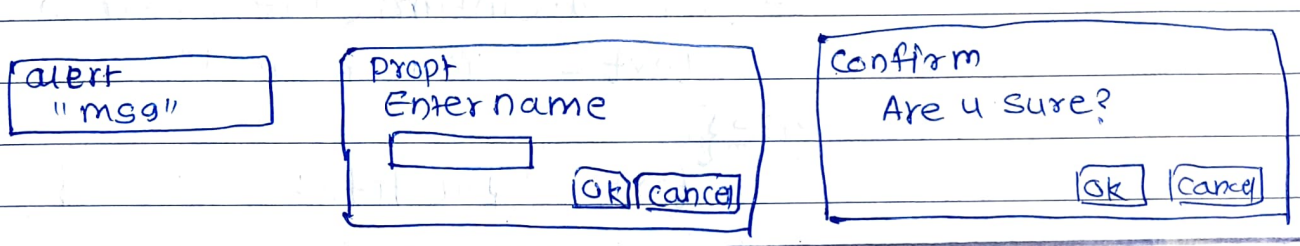
JS popup Alert

Page No.: YOUVA
Date:

Javascript has three kind of popup boxes
1) Alert box 2) Confirm box 3) prompt box

1) window.alert() - popup alert msg.
2) window.prompt() - Popup msg with input and submit/OK
3) window.confirm() - Popup confirm msg with OK Cancel option

Postly




```
<body>
  <P id = "demo" ></P>
  <button onclick = "myfun1()" > alert </button>

  <button onclick = "myfun2()" > confim </button>

  <button onclick = "myfun3()" > prompt box </ button>

</body>

<script>
// function for alert.
      function myfun1 (){

          window.alert ("This is alert");
            }

// function for Propt.
      function myfun2() {

          var text;
          var person = window.propt("Enter Name");
          if (person==null || Person == ""){
                text = "user cancelled propt";
```

```
else {
  text = " Hellow" + person + " How r U? ";
  }
document.getElementById("demo").innerHTML = text;
}
//function for confirm.
    function myfun3() {
    var text;
    if (window.confirm("press a button")) {
        text = " You confirmed";
    else {
        text = " you didn't confirmed";
      }
document.getElementById("demo").innerHTML = text;
```

JS Timing

window object allows execution of code at specified time intervals.

these time intervals are called timing events

two key methods to use with Javascript are:

1) setTimeout(function, milliseconds)
Execute function after specified interval of time.

2) setInterval(function, milliseconds)
Repeat the execution of function after interval of time

3) clearTimeout(var holding setTimout() function)
Stops and clears execution of setTimeout() function

4) clearInterval(var holding setInterval() function)
Stops and clears execution of setInterval() function.

eg.

```
<button onclick="myVar = setTimeout(myfun, 3000)">start</button>
<button onclick="clearTimeout(myVar)">stop</button>
<button onclick="myVar2 = setInterval(myfun2, 3000)">start</button>
<button onclick="clearInterval(myfun2)">stop</button>

myfunction myfun(){
    var d = new Date();
    document.getElementById("demo").innerHTML
        = d.toLocaleTimeString();
}
```

```
function myfun2(){

    var d = new Date();
    document.getElementById("demo").innerHTML
            = d.toLocaleTimeString();

}
```

JS Cookies

Cookies are data, stored in small text files, on your computer.
When web server sent a web page to browser, the connection is shut down, and the server forgets everything about user.
Cookies are invented to solve the problem.
- i) When user visits webpage, his/her name can be stored in cookies.
- ii) next time the user visits page the cookie "remembers" name of user.

Cookies saved in name-value pairs like-

`username = John Doe`

When browser requests a webpage from server, cookies belonging to the page are added to the request
This way the server gets the necessary data to remember information about users.

1) Create cookie with JS

```
document.cookie = "username = John ;
                   expires = Thu, 18 Dec 2013 12:00:00 UTC ;
                   Path = / ";
```

2) Read cookie with JS

```
Var x = document.Cookie ;
```

these will return all the cookies in single line String

3) Change a cookie with JS

you can change a cookie the same way we create it. (Overwrite).

```
document.cookie = "username = Shubham;
                   Path = / ";
```

4) Delete a cookie with JS

To delete cookie don't specify cookie value when you delete a cookie, just set expire parameter to passed date.

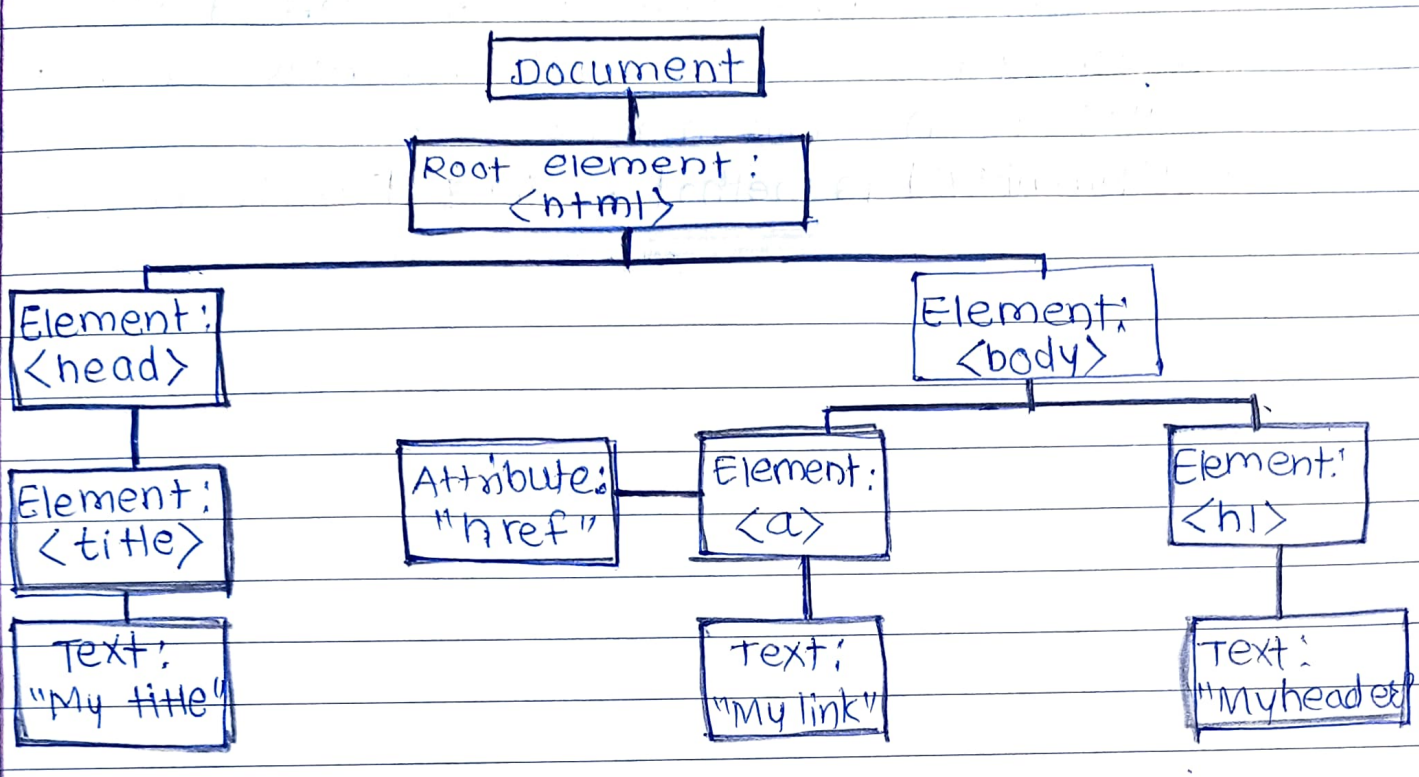
```
document.cookie = "username = ;
                  expires = Thu, 01 Jan 1970
                            00:00:00 UTC;
                  Path = /; ";
```

Functions set/get/check cookies.

JS HTML DOM

DOM - Document object model.
When webpage is loaded browser creates Dom of page.

HTML DOM model constructed as a tree of objects.



With object model, JS gets all power it needs to create dynamic HTML :-

1) JS can create, add, change and remove the HTML elements and attributes in the page.
2) JS can change all the styles in the page.
3) JS can react to all existing HTML events and can create new events.

What is DOM -

DOM defines standerd for accessing documents. It allows program and scripts to dynamically access and update the content, structure and style of a documents. It has three parts :-

1) Core DOM → for all documents
2) XML DOM → for xml docs
3) HTML DOM → HTML Docs

DOM Programming interface

* HTML DOM methods are action performed on HTML elements and properties are values that we can set or change for elements.

* In DOM all HTML elements are defined as objects which have methods and properties which togeth called DOM programming interface.
getElementById is method, innerHTML is property.

JS Document Object.

The document object represent webpage hence any HTML element is accessed by through document object.

JS Dom Element

Page No.:
Date:
YOUVA

1) Finding Element by id

```
Var x = document.getElementById("id")
```

If element found, it will be return the element as an object otherwise x will be null.

2) Finding Element by Tag Name.

multiple elements are returned hence [Element]

```
Var x = document.getElementsByTagName("p")
```

3) Finding Element by class Name

retur all elements with classname [intro] applied

```
Var x = document.getElementsByClassName("intro")
```

4) Finding Element by CSS sectors.

```
Var x = document.querySelectorAll("css sector")
```

any CSS selectors one or more separated by [,] are used. like - P, P.intro, .intro

5) Finding Element by Object Collections

```
Var x = document.forms["form1"]
```

All the elements inside object collection is selected and returned to variable as list.

Otherobject collections are -

1) document.anchors
2) document.body
3) document.documentElement.

4) document. embeds
5) document. forms
6) document. head
7) document. images
8) document. links
9) document. scripts
10) document. title

JS - Changing HTML

1) HTML output stream
used to write HTML content dynamically

```
document.write("content")
```

2) Changing HTML content

```
document.getElementById("id").innerHTML="";
```

OR

```
var x = document.getElementById("id")
x.innerHTML = "new HTML content"
```

We use **innerHTML** method to change the content of element retrieved by getter.

3) Changing Value of Attribute

```
document.getElementById("img").src = "picture
```

We retrieve element with getter and with .att we set value of attribute to new value.

JS - Changing CSS

1) Changing HTML Style

```
document.getElementById("id").style.color = "red"
```

we retrieve element then with `.style` we access styling properties and `.Property` we assign value to that property.

JS Events

1) Mouse click - `onclick`

i) `<h1 onclick="this.innerHTML='Oops!'">Click here</h1>`

`onclick` attribute defines action to element
`this` defines id as the same element which specified.
`innerHTML` changes content.

ii) `<h1 onclick="changeText(this)">Click here</h1>`

```
<script>
    function changeText(this){
        id.innerHTML='Oops!';
    }
</script>
```

`changeText( )` is function
`this` used as id argument inside function

iii) `<h1 onclick="myFunction()">Try it!</h1>`

iv) Assign Event using HTML DOM

```
<script>
    document.getElementById("myBtn").onclick=displayDate;
</script>
```

`displayDate` is a function assigned.

2) Onloading and unloading webpage.

onload - Triggered when webpage loaded

onunload - Triggered when webpage unloaded.

3) onchange of input fields

onchange

e.g.

```
<input type="text" id="fname" onchange="Upper()">

<script>
    function Upper(){
        Var x = document.getElementById("fname");
        x.Value = x.value.toUpperCase(); }
</script>
```

output

when we insert any lowercase letter to text and left text field they converted to Upper case.

4) Moving mouse over Element

onmouseover - Triggers by hovering mouse over elem

on mouseout - Trigers by hovering out mouse.

5) Stroking a mouse key

onmousedown - Triggers when mouse clicked and ho

onmouseup - Triggers when mouse key released

6) on focusing to input field

onfocus

e.g.

```
<input type="text" onfocus="function(this)">

<script>
        function function(id){
            id.style.background = "yellow";
            }
</script>
```

Output

on clicking into text box, textbox becomes yellow colo

JS Event listener

M T W T F S S
Page No.:
Date:
YOUVA

Event listeners

addEventListener()

1) addEventListener() method add event handlers without overwritting existing event handlers.
2) We can add many event handlers by many event listeners.

Sytanx

```
element.addEventListener(event, function, usecapture)
```

element - element revived by getter method/element
event - events like onclick, onmouseup etc

function - function to triggere on event

usecapture - The values are true / false

usecapture is either Event Bubbling or EventCapturing
e.g. when has <P> inside it both have events then -
Event Bubbling - when value set to false (defaut) inner element is triggered first that event to inner handled first then event of outer handled.

Event capturing - when value set to true outer element's event handled first then inner element's event handled.

e.g.

```
<body>
  <div>
      <p>  my text  </p>
  </div>
</body>

  <script>

document.getElementByTagName(div).addEventListener("click", myfu
                                                              tr

document.getElement By Tag Name(p).addEventlistener("click", myfunct
                                                              tru

  function myfunction(){
      alert("Hello world-1);
    }

  function myfunction2(){
      alert("Hello world-2);
    }

  </script>
</body>
```

output

When we keep ~~true~~ event handler then clicking on "my text" propts Helloworld-1 first then Helloworld-2

But keeping it default/false it prompts Hello world-2 first then Hello world-1

Event Listeners to Window object.

Window objects also supports Eventlisteners.
e.g.

```
window.addEventlistener("resize", function(){
    document.getElementById("demo")
    .innerHTML = "text";
});
```

Remove Event Listener

`removeEventListener()`

1) It removes event handlers that have been attached with the `addEventListener()` method.
e.g.

```
element.removeEventListener("mousemove" myfunction)
```

By moving mouse event handler to element is removed.

JS DOM Nodes

M T W T F S S
Page No.:
Date:
YOUVA

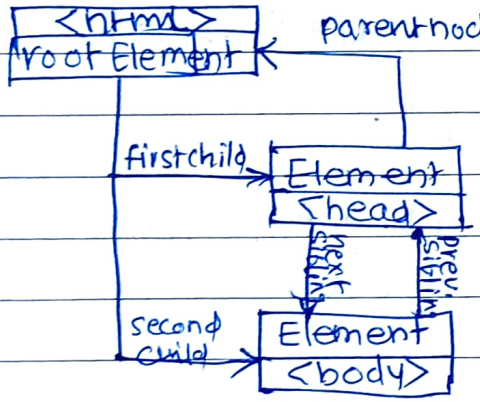
Everything in HTML document is a node.

1) Entire document is document node.
2) HTML element is an element node.
3) text inside HTML element are text node.
4) HTML attributes are attribute node.
5) All comments are comment node.

Nodes can be created, modified, deleted or accessed by JS.

Node Relationship

Nodes in the tree node have hierarchial relationship to each other.



Every node have parent Except root node.
Nodes have no. of childs who are siblings of each other

Creating New HTML Elements (Nodes)

For creating HTML element, we create the elem first and append it to existing element.

Example

```
<div id="div1">
    <p id="p1">This is paragraph</p>
    <p id="p2">This is another pargraph</p>
</div>

<script>
    Var para = document.createElement("p");

    Var node = document.createTextNode("This isne

    para.appendChild(node);

    Var element = document.getElementById("divl
    element.appendChild(para);

</script>
```

Methods

1) document.createElement("Element")

2) document.createTextNode("Textdata")

3) element.appendChild("node")

Remove Existing HTML Elements.

To remove HTML elements ~~by~~ use remove() method.

```
<div id = "div1">
    <p id = "p1"> This is a paragraph </p>
    <p id = "p2"> This is a another paragraph </p>
</div>
<button onclick = "myfunction()" > Remove Element </button>
<script>
    function myfunction(){
        var element = document.getElementById( );

        element.remove();
    }
</script>
```

Methods

1) element.remove()
remove element assigned to element variable.

2) parent.removeChild(child)
parent variable is assigned by Parent Element (div)
child variable is assigned by child Element (p)
method removes child from parent.

3) child.parentNode.removeChild(child)
child is variable to which child node Element assigned
parentNode returns parent of child.

Replace HTML Elements

```
<div id ="div1"
    <p id = "p1"> This is paragraph </p>
    <p id = "p2"> This is another Paragraph</p>
</div>

<script>
Var  para = document. CreateElement ("P" ) ;
Var  node = document. create TextNode("This is new"
para.appendChild (node);

Var parent = document. getElementById ("div1");
Var child  = document. getElementBy Id (" p1" );
parent. replace Child (para , child );

</script>
```

Methods

1) parent . replaceChild (old Element variable , new Element

Parent is variable to which parent node assigned
Old Element Variable and new Element Variable
variables to which old element and new element
assigned.

JS DOM HTML DOM Collections

M T W T F S S
Page No.:
Date:
YOUVA

The `getElementsByTagName()` method returns an `HTMLCollection` object

This HTMLCollection object is array-like list (collection) of HTML elements

Example

```
Var x = document.getElementsByTagName("p");
Var y = x.length;
```

this returns collection of all `p` element and assigne to `x` variable

`y = x[1];` we can access elements by index.

Collection is array like object but not an array
we cannot use methods of array on HTMLcollection

JS DOM HTML DOM NodeLists

The [NodeList] is object of list of nodes extra... from a document. NodeList object is returned b... [childNodes] or [querySelectorAll()]

Examples

```
Var x = document.querySelectorAll("p");
Var y = x.length;
Var z = x[0];
```

Difference Between HTMLCollection and NodeLi...

1) [HTMLCollection] is collection of HTML elem...
2) [NodeList] is collection of document No...
3) Both of them are array like list with leng... index accessibility property.