02: Introduction to JS

⇒ Introduction of JS =

* JS is client-side object based scripting lang used today on the web.
* It gives -
  - create pop-up windows.
  - manipulate txt
  - perform mathematical cal on data.
  - Retrieve the crnt date & time frm a user's comp.
  - Determine the user's screen size, window version, screen resolution.

→ Script in webpgs =

* primary method to include JS within HTML is <script> elemnt.
* A script brousr assumes that all the txt within the <script> tag is to be interpreted as some form of scripting lang, by dfault it is JS.
* eg = <script language = "javascript">
  -- --
  </script>
  (or)
  <script type = "text/javascript">
  ~ ~
  </script>.

* There are 3 places you can put ur JS -
  1) In head
  2) In body
  3) In ex file.

I In <head> element =

* You can place <script> elemnt inside the <head> elemnt of Html doc.
* The script placed inside <head> elemnt runs whn you perform some action.
* eg :-
```
<html>
  <head>
    <title> Hi </title>
    <script>
      alert ("welcome");
    </script> </head> </html>
```

II In <body> section =

* Script elemnt placed inside <body> elemnt ~~of~~ runs whn a webpg start loading in a web brousr.
* eg :-
```
<body>
  <script>
    alert("welcome");
  </script>
</body>
```

III) In ex. file =

* You can store the JS code in an ex file & save that file using .js extension.
* Nxt, you need to link the ex file with Html doc by using src attribute of script elemnt.
* eg→ hello.js:

```
function Hi() {
        alert("welcome");
    }
```

html pg:

```
<body>
<button type="button" onclick="Hi()"> clickme </button>
</body>.
```

→ Script hiding =

* 1 way to mask JS is to use Html cmnts around the script code.
* eg :-

```
<script>
  <!--
      put yr JS code here
  // -->
</script>
```

→ <noscript> elemnt =

* It is used when a browsr does not support JS / JS is turned off.
* eg :-

```
<body>
  <script>
      alert("welcome");
  </script>.
  <noscript>
    <em> your browsr does not sprt JS </em>.
  </noscript>.
</body>.
```

→ Comments =

They are txt within yr script that is ignored by the browsr when the script runs.

a) Single-line cmnts:

eg → var x=5 ; // declare x.

b) multi-line cmnts:

eg → /* following code ... */.

→ variables =

* used to store data like str or no., txt, etc.
* data stored in the variable can be set, updated, retrived whneur needed.
* You can create a variable using var keyword & assignmnt operator (=).
* eg → var name = "Ansar";
var age = 21;
var area = 2.5;
var married = false;
} varname="Ansar", age=21, area=2.5;
* var is optional. → age = 21;
* ① <script>
var username;
username = "Ansar";
document.write(username);
</script>
→ Ansar.
* ~~Constnt~~ ES6 introduces 2 new keywords- let & const.
let = var.
* ② <script>
let name = "Ak";
let age = 11;
let isStdnt = true;
doc.write(name + "
");
" age
" isStdnt

const pI = 3.14;
doc.write(pI); </script>.

* for loop =
<script>
for(let i=0; i<5; i++) {
doc.write(i + " ");
}
→ 0 1 2 3 4
doc.write("<hr>" + i);
→ undefined
for(var i=0; i<5; i++) {
~~doc.write("<hr>" + i);~~
doc.write(i + " ");
→ 0 1 2 3 4
}
doc.write("<hr>" + i);
→ 5

- <hr> → horizontal rouler → ———
- var → functional scoped variable (created using ();.).
- let & const → block level scoped. (new scope is created b/w a pair of {}).

→ Datatypes =

* Specify wht kind of data can be stored & manipulated within a prgm.
* 3 grps -

a) primitive (D) → Str, No., boolean.
b) composite (D) → obj, array, ().
c) Sple (D) → undefined, null.

~~I Str (D)~~ primitive (D) can hold only 1 value at a time whereas composite (D) can hold collections of values.

I Str (D) =

```
var a = 'Hi' ;   // Single quote
var b = "Hi" ;   // double quote.
doc. write (a + "<br>");
doc. write (b);
```

II Number (D) =

*
```
var a = 25 ;
var b = 80.5;
var c = 4.25e+6;
var d = 4.25e-6;
```
exp

* doc. write (16/0 + "
"); → Infinity
" " (-16/0 + "
"); → -Infinity
" " (16/-0); → -Infinity

• Infinity → represnts mathematical op.

* NaN → represnts a Sple not-a-number value.
eg → doc. write ("Some txt" /2);
→ NaN.
doc. write (math.sqrt(-1));
→ NaN.

III Boolean (D) =

*
```
var isready = true;
var issleep = false;
d. w (isready + "<br>")   → true.
d. w (issleep);           → false.
```

*
```
var a=2 , b=5 , c=10;
alert (b>a)   → true
alert (b>c)   → false.
```

IV Object (D) =

Complex (D) that allows you to store collection of data,
obj contains proprties as a key:value pair.

* var emptyobj = {};
var person = { "name": "AK", "age": 10 };
var car = {
"model": "BMW",
"color": "red"
}
consde.log(person);

V Array (O) =
* type of obj used for storing multiple values in single variable.
* var colors = ["red", "yellow"];
alert(colors[0]); → red.

VI function (O) =
* It is callable obj that executes a blk of code.
* var gr = function () {
return "Hello";
}
d.w(typeof gr); → function
d.w(gr()); → Hello.

VII Undefined (P) =
* It can only have 1 value — the spl value • undefined.

* var • a;
var • b = "Hello";
d.w(a); → undefined.
d.w(b); → Hello.

VIII Null (P) =
* Anthr spl O that can have only 1 value — the null value.
* var a = null;
d.w(a); → null.
var b = "Hi";
var b = null;
d.w(b); → null.

→ JS operators =

Symbols that tells the js engine to perform some sort of actions.

I Arithmetic (O):
var x = 10, y = 4;
d.w(x+y); → 14
d.w(x-y); → 6
d.w(x*y); → 40
d.w(x/y); → 2.5
d.w(x%y); → 2

II Assignmnt (o) :

```
var x;
x = 20;
x += 30;
d.w(x); → 50.
x *= 50;
x -= 20;
d.w(x); → 30.
x = 5;
x *= 25
d.w(x); → 125
x = 50;
x /= 10;
d.w(x); → 5
x = 100;
x %= 15;
d.w(x); → 10
```

III String (o) =

① concatenation
② concatination assignmnt.

```
eg→ var s1 = "Hello";
    var s2 = "world";
    d.w(s1+s2+"<br>"); → Hellcworld
    s1 += s2;
    d.w(s1); → Hello world.
```

IV Incremnt / Decremnt (o) :

```
var x;
x = 10;
d.w(x); → 10
d.w(++x); → 11
d.w(x); → 11
x = 10;
d.w(x); → 10
d.w(x++); → 10; d.w(x) → 11
x = 10;
d.w(x--); → 10; d.w(x) → 9
d.w(--x); → 9; d.w(x) → 9
```

(*) ++x (pre incremnt) → incremnts x by 1, then return x

(*) x++ (post incremnt) → returns x, then incremnt x by 1.

(*) --x (pre decremnt) → decremnts x by 1, then return x.

(*) x-- (post decremnt) → returns x, then decremnt x by 1.

V Logical operators :

&& → And → x && y
|| → or → x || y
! → Not → !x (x is not T)

[used to combine cond^n (s).]

```
var yr = 2018;
if ((yr % 400 == 0) || ((yr % 100 != 0) &&
    (yr % 4 == 0)))
{
    d.w (yr + " leap year");
}else {
    d.w (yr + " not leap year");
}
```

VI Comparison (6):

used to combare 2 values in a boolean fashion.

*	==	equal	x==y
*	===	identical	x===y
*	!=	Not equal	x!=y
*	!==	not identical	x!==y
*	<	-	-
*	>	-	-
*	<=	-	-
*	>=	-	-

```
var x = 25, y = 35, z = "25";
d.w (x==z);    → T
d.w (x===z);   → F
d.w (x!=y);    → T
d.w (x!==y);   → T
d.w (x<y);     → T
d.w (x>=y);    → F
```

→ JS conditional / selection (5):

I if & if...else (5):

```
var now = new Date();   // we get date & time
var dayofweek = now.getDay(); → 0-6 → sun-sat
if (dayofweek == 5) {
    d.w ("nice week");
}else {
    d.w ("nice day");
}
```

getday
sun → 0
mon → 1
sat → 6

getDate() → returns weekday as a num from 0-6
getDay → returning the day.

II Switch (5):

```
var d = new Date();
switch (d.getDay()) {
    case 0:
        d.w ("sun");
        break;
    case 1:
        d.w ("mon");
        break;
    ;
    default:
        d.w ("no info");
}
```

* multiple selection statement

III Ternary (o) :

* Syn → var Rslt = (condn) ? val1 : val2;
* provides a shorthand way of writing the if-else (s).
* var age = 21;
  var user = age < 18 ? 'child' : 'adult';
  d.w (user);
  → Adult

Selection (s)

nested if | if | if-else | if-elseif | Switch | Ternary

→ Looping statements :

for | while | do-while | for-in | For-of

entry control loop | exit c.loop | for-in → works in obj | For-of → works in [], str

I while loop =

Syn → while (condn) { (s) }

Syn → initialization
while (condn) {
(s)
updation;
}

eg → var i = 1;
while (i <= 5) {
d.w ("number is:" + i + i++);
}
→ number is : 1
" : 2
⋮
" : 5

II Do-while =

eg →
var i = 1;
do {
d.w ("number is:" + i + i++);
}
while (i <= 5);

III for loop =

eg →
<script>
for (i = 1; i <= 5; i++) {
d.w ("number is:" + i + "
");
}
</script>

Syn → for (initialization; condn-, updation)
{
// (s)
}

IV for... in =

str → for (variable in obj) {
// body of loop
}

variable → str value.

eg → <script>
// an obj with some pro-
var person = {"name":"amar", "surname": "akbar", "age": 25};

for obj iteration {
for (var pro in person) {
d.w (pro + "=" + person[pro] + "
");
}

→ name = amar
surname = akbar
age = 25

</script>

for array

<script>
var f = ["App", "Bay", "man"]
for (var i in f) {
d.w (f[i] + "
");
}
</script>

→ App
Bay
man

for...in → only str use, so better option is to use for loop for array.

V for... of =

eg → <script>

iterating over array

let letr = ["a", "b", "c"];
for (let ll~~et~~ of letr) {
d.w (ll~~et~~ + ",");
}

d.w ("<hr>");

iterating over str

let grt = "Hello";
for (let char of grt) {
d.w (char + ",");
}

</script> → a, b, c

H, e, l, l, o

→ Jumping statemnts = (break, continue)

I break:

// JS loop through an array using for-in loop.

for (var i=0; i<5; i++) {
if (i==3) {
break;
}
d.w ("no. is after executing break is " + i + "
");
}
d.w ("the no. is :" + i);

→ the no. is after executing break is 3

the no. is 0
" 1
" 2

II continue :

```
for(var i=0; i<5; i++) {
    if (i==3) {continue;}
    document.write("no. is :" +i);
}
```

→ no. is 0
" 1
" 2
" 4

⇒ JS output =

I console.log() :

// writing into browser's console with JS

```
<script>
    console.log("Hello");
    var x=10, y=5;
    var sum = x+y;
    console.log(sum);
</script>
```

→ Hello
15

II write() & writeln() :

```
<script>
    var x=10, y=5;
    var sum = x+y
    doc.write(sum) → 15
    doc.writeln(sum) → 15
```

Note :

If you use doc.write() method after the pg has been loaded, it will overwrite all the existing content in that doc. (prev data erase ayyindi)

[writeln → creates a new line after each line but if we are using write(), we use
]

eg for write() –

```
<html>
<body>
<h1>This Heading</h1>
<p>This is the para of txt</p>
<button type="button" onclick=
    document.write("Hello Ansar")>
    click me</button>.
</body>
</html>
```

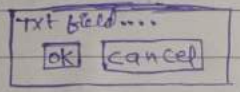
→ Heading
This is the para of txt
click me

(prev data erases)

Hello Ansar

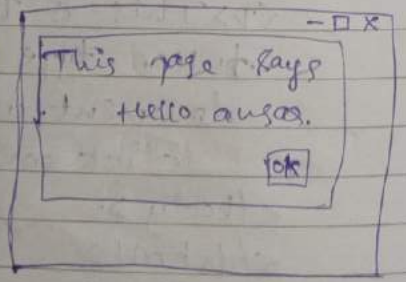
doc.write/ln() method is similar to write(), only it adds a new line char after each statemnt.

⇒ JS Dialog/pop-up box :



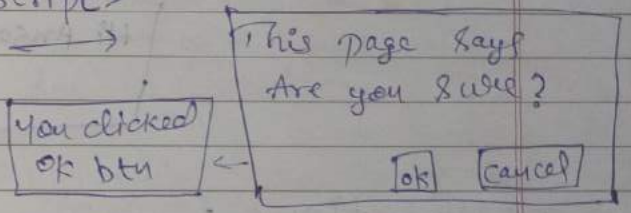
1) alert box : → only ok button
used to display an alert msg using an JS code.

eg:-
```
<script lang="js" type="txt/js">
var ak = "Hello answer";
alert(ak);
alert("Another alert box");
</script>
```



2) Confirm box : → ok & cancel
used to allows a user to confirm / cancel an action.

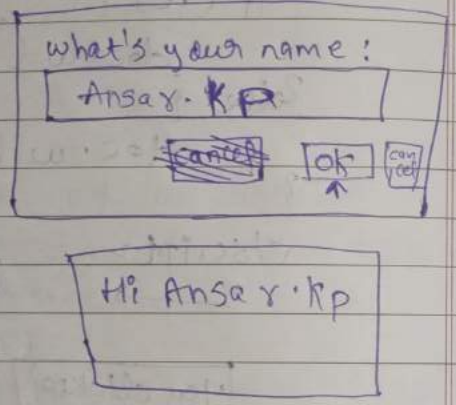
eg:-
```
<script>
var re = confirm("Are you sure?");
if(re){
    doc.w("you clicked ok btn");
}else{
    doc.w("you clicked cancel btn");
}
</script>
```



3) prompt box : → I/p & ok, cancel.
used to prompt the user to I/p something. & hve ok & cancel btns.

eg:-
```
<script>
var n = prompt("what's your name!");
```

```
if(n.length>0 && n!="NULL") {
        doc.w("Hi" + n);
} else {
        doc.w("Invalid name");
}
</script>
```



⇒ JS functions :

* It is a block of (s) that execute a specific task.
* Code reusability
* function (keyword) → declare, def, call.
* eg:- <body>
<script>

```
function sayHello() {
        doc.w("welcome to Ak world");
}
</script>
sayHello();
</script> </body>
```

I Adding paramtr to the ()s :

eg:

```
<script>
function sum(a,b) {
        var total = a+b;
        doc.w(total);
}
sum(10,20);          → 30
d.w("<br>");
sum(1,2);            → 3
sum(-5);             [a=-5 & b=undefined : → NAN]
</script>
```

II Return (8) frm function :

```
<script>
function getsum(n1,n2) {
        var total = n1+n2;
        return total;
}
```

// displaying return value
d.w (getSum (6,20)); → 26.
d.w ("
");
d.w (getSum (-5, 17)) → 12

</script>

→ scope of variable = refers to an area within which a () & its variables are accessible

Global:	Local:
~~all area in a pg~~	~~inside a f()~~
Global ()s: alert (), prompt box, confirm box, eval () (evaluat str), isFinite () (non finite nos/infinite T/F)	
* a () that can be accessed anywhere in a prgm.	* can be accessed only within a specific area of the prgm.

→ JS Events :

* Action performed by the user.
* It is smthing that hpns when user interact with the webpg.
  - → when clicked a link / btn
  - made selection in a select box
  - pressed key on the keyboard
  - moved the mouse pointer.

* events are categorized into 4—
  - mouse (e)
  - keyboard (e)
  - form (e)
  - doc/window (e)

1. mouse events =

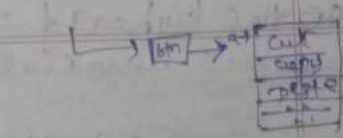
a) onclick (click evnt): [occurs when a user clicks on an elemnt on a web pg.]

```
<body>
<button type="button" onclick="alert('you hve clicked a btn');"> click me
</button>
<a href="#" onclick="alert('clicked a link');"> click </a>.
<body>.
```
(evnt handler is onclick).

b) oncontextmenu (contextmenu evnt):
occurs when user clicks the right mouse btn on an elemnt to open context menu.

```
<button type="button" oncontextmenu="
"alert('you hve right clicked on btn');">
right click me </button>.
```

* evnt handler is oncontextmenu.



(btn → over → a.)

c) onmouseover =
occurs whn a user moves the mouse pointer over an elemnt.

d) onmouseout = (btn → → alert.)
occurs whn a user moves the mouse pointer outside of an elemnt.

II keyboard evnts :
This evnt is fired whn the user press / release a key on the keyboard.

a) onkeydown : (b → alert → b)
occurs whn user presses down a key on the keyboard.
<input type="text" onkeydown="alert('you pressed a key');">

b) onkeyup : (b → b → alert)
whn user releases a key on the keyboard.

c) onkeypress :
occurs when a user presses down a key on the keyboard that has a character value associated with it.

III Form evnts :
It is fired whn a form control receive / loses focus / whn the user modify a form control value.

a) onfocus :
occurs whn the user gives focus to an elemnt on a webpg.
<input type="text" onfocus sample(this)">
<button type="button"> click </button>.
<script>
function sample (el) {
el.style.background = "green";
}
</script>.
* 'this' keyword inside an evnt handler refers to the elemnt where the evnt is currently being delivered.

b) onblur :
occurs whn the user takes the focus away frm a form elemnt / a window.
<input type="text" onblur="alert('inputbox')">
<button type="button"> Submit </button>

Submit
1st click inside the input box then click outside to see the alert box.

c) onchange :

occurs when a user changes the value of a form element.

```
<select onchange="alert('changed');">
  <option> male </option>
  <option> female </option>
</select>
```

→ select any option in select box to see alert box.

d) onsubmit :

occurs when user submits a form on a web pg.

```
<form method="post" onsubmit="alert('
    Submitted form');">
  name: <input type="text">
  <input type="submit" value="Submit">
</form>
```

→ [ ] → Submit → alert box.

* <form action="..." load (...) ... on background">

* <form method="post" ...> for secure.

IV Doc / window event :

Events are also triggered when the pgs has loaded / when user resize the browser window, etc.

a) onload :

load event occurs when a web pg has finished loading in the web browser.

event handler is onload event handler.

eg: <body onload="alert('onload event');">
<h1> this is heading </h1>
 This is paragraph
</body>

→ pg load → alert → <h1> & .

b) onresize :

occurs when a user resizes the browser window.

Also occurs in the situations when the browser window is minimised / maximised.

event handler is onresize.

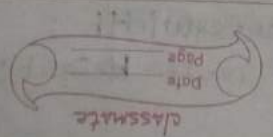
eg:-

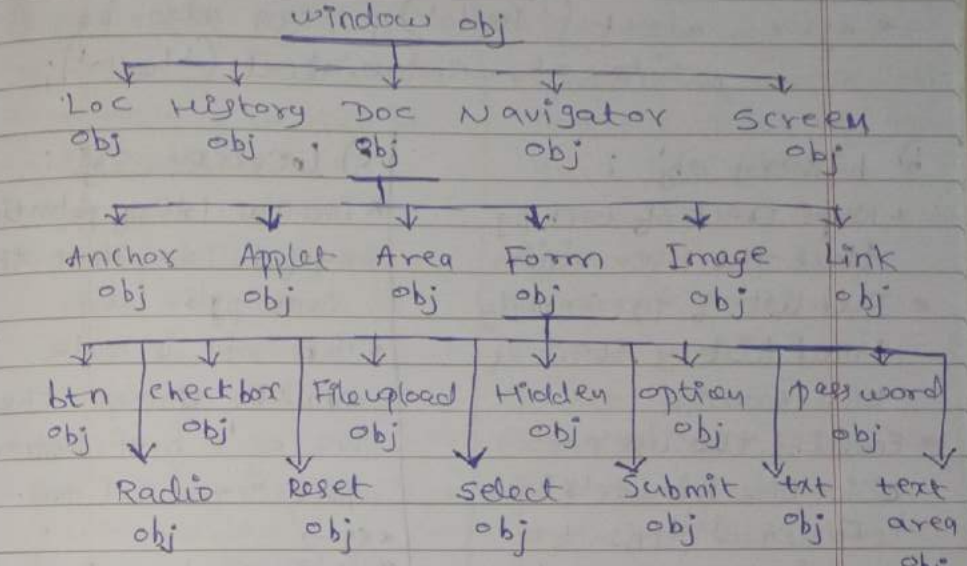
```
<script>
  function sample() {
    var w = window.outerwidth;
    var h = window.outerheight;
```

```
var = " width = " + w + "height = " + h;
doc.getElementById('res').
    innerHTML = txt;
},
window.onresize = sample;
</script>
```

=> DOM :

* DOM is a cross platform & lang independent interface that allows prgms & scripts to dynamically access & update the content, str & style.
* DOM is a web standard defined by W3C
* DOM is separated into 3 parts : Core, HTML & XML.
* Core provides a low-level set of obj that can represented any structured doc.
* HTML & XML provides additional, higher-level interfaces that are used with the core specifications.

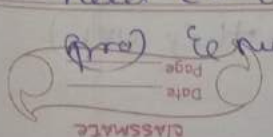




I DOM objects =

a) window obj :

* Represnts the browser's frame / window in which yr web pg is contained.
* Also represnts (pro) of window obj
  - you can find out wht browsr is running
  - pgs the user has visited.
  - Size of browsr window.
* It is a global obj, means you don't need to use its name to access its (pro) & methods.



* eg → alert ('hello'); can also be write as window.alert ('hello');

b) history obj :

* keeps track of each pg that the user visits.
* This list of pgs commonly called history stack of the browsr.
* Enables the user to click the browsr's Back & Forward btns to revisit pgs.
* history obj has back () & forward().
* go() → takes 1 paramtr that specifies hw far forward (backward in the history stack you want to go.
* eg → ① history.go(-2); (return user to the pg before the previous pg).
history.go (3); (go forward 3 pges).
② history.length → 1st I visit youtube then fb then google (now I visited 3 websites) ⟹ 3 o/p.
③ h.back() → load previous URL
④ h.forward() → load next URL

c) location obj :

* contains lots of potentially useful info about the crnt pg's loc.
* This info is made available through the loc obj's href, hostname, port & protocol (pro).
* eg →
① window.location = "https:..."; (pg load [illegible])
② doc.write (loc.href); (returns crnt pg URL)
③ doc.w (loc.hostname); (returns hostname of crnt pg. 127.0.0.1...)
④ doc.w (loc.protocol). (returns crnt web protocol pg)
⑤ loc.reload (); (continue the reload op^r)
⑥ loc.replace ("https:..."); (crnt URL is replaced with this URL)

d) navigator obj :

* Anthr obj that is a property of window & is available in all browsrs.
* contains lots of info about the browsr & the OS in which it is running.
* eg →
① doc.w (navigator.cookieEnabled); (if cookie is enable return T otherwise F)
② nav.appName
③ nav.appVersion
④ nav.appCodeName
⑤ nav.platform
⑥ nav.language

e) Screen obj :

* property of window obj contains a lot of info about the display capabilities of client machine
* eg →
① doc.w (screen.height);
② screen.width;
③ s.colorDepth;
④ s.pixelDepth;
[returns colordepth in user's settings & pixel depth]

f) document obj :

* 1 of the most imp & commonly used obj in the DOM.
* Represents the whole doc.
* You can gain access to the HTML elements, their (pro), & methods inside yr pg.

* forms collection contains all the <form> tags in the doc.
* images collection represents all the images in a doc.
* links collection represents all the hyperlinks within a pg.
* anchors collection represents all the anchors in a doc.

II properties, methods & obj =

* Each obj can have properties & methods.
* property tells you something about an obj & a method performs an action.

1] properties, & methods of doc obj :

* Following table are the properties of doc obj -
* If you can set a property, it is → read/w property, whereas the ones you can only read are → read-only.

properties	Description
1) Cookie	returns all name/value pairs of cookies in the doc.
2) docMode	returns the mode used by the browser to render (in browsers) the doc.
3) Domain	returns the domain name of the svr that loaded the doc.
4) lastModified	returns the date & time of last modified doc.
5) readyState	returns the status of doc.
6) Title	returns the title of doc
7) URL	returns the full URL of doc.

eg:- document.title → to access title

* Methods perform actions & are always written followed by a pair of brackets with optional list of parameters.
* methods of doc obj are -

methods	Description
1) close()	closes the o/p stream previously opened with doc.open().
2) clear()	clears the doc in a window.
3) getElementById()	Access the 1st element with the specified id.
4) getElementByName()	-"- name.
5) getElementsByTagName()	-"- tagname.

6) open()
7) write() & writeln()
↓ writes o/p to a doc ↓ adds a new line char aftr each statemnt.

* eg:-

```
function check() {
var username = doc.getElemntById("uname");
var password = "                    ("pass");
if(username.value == "abc" &&
    password.value == "123")
        alert("Successfully Signed in");
else
        alert("Invalid");
}
<body>
username : <input type="txt" id="uname">
password :  "    type="password"
                 id="pass">
<input type="button" onclick="check()"
    value="Sign in">
</body>
```

2] Proprties & methods of forms collection :

methods of form obj—

reset() — reset all form elemnts to their default values.

submit() — submits the form.

* directly access that form obj using its name—
doc.frmLogin.action.
└ varname.

* proprties of form obj—

1) action	returns value of the action attribute in a form.
2) length	returns no. of elemnts in form.
3) method	returns the value of the method attribute in form.
4) name	returns value of the name attribute in form.
5) noValidate	returns whether the form-data should be validated (not.
6) target	returns the value of the target attribute in form.

3] properties & methods of form elemnts:

* Each <form> has an elements[] collection obj as a proprty which reprsnts all of the elemnts in that form.
* elemnts in a form—
a) txt fields
b) checkboxes & radio btn.
c) buttons
d) select boxes.

* properties of form elemnts —

1) checked (checkboxes & radio btn)	returns T whn checked	false whn not.
2) disabled (All elemnts)	returns T whn disabled & user cannot interact with it.
3) form (All E)	returns a refrnce to the form it is part of.
4) length (Select box)	no of options in <select> elemnt.
5) name (All E)	Access the name attribute of the elemnt.
6) type (All E)	returns type of form control

* methods of form elemnts :-

1) blur()	Takes focus away frm currently active elemnt.
2) click()	user's clicking the mouse ovr the elemnt.
3) focus()	gives focus to the elemnt.
4) select()	selects the txt in the elemnt.

* eg:-

```
<body>
 <form name="frmLogin" onsubmit=
    "alert ("hello" + document.frmLogin.
    txtUsername.value)" id="frmLogin">
 username : <input type="txt" name="txtuname"
      size="12" />
 <input type="submit" value="click here">
 </form>
```

⇒ Built-in obj =

I String obj :

* Str obj allows you to deal with str of txt.
* To create an instnce of Str obj —

```
mystr = new String('Hai');
alert (mystr.length);
```

* Property of str obj :

length — returns the no. of chars in a str.

* methods of Str obj :

1) anchor(name)	creates an <a> elemnt.
2) big()	Displays txt as it is in a <big> elemnt.
3) bold()	
4) fixed()	Displays txt as it is in a <tt> elemnt.

5) fontcolor(color)	displays txt as if in a <font> elemnt with a color attribute.
6) fontsize(fontsize)	
7) sub()	displays txt as if in a <sub>.
8) toLowerCase()	converts a str to lowercase.

eg:- <script>
mystr = new String ('learn obj easy');
mystr = mystr.substring (15, 31)
(start position, end pos)
mystr = mystring.toUpperCase()
doc.write (mystr);
</script>

II Date obj = helps you wrk with dates & times.

* you can create new date obj using date constructor ~~+~~ new Date()
* eg:- var bdate = new Date ("April 16, 1975");
* eg:- var today = new Date()
var newyr = new Date (2011, 11, 31)
var d_remaining = (newyr - today)
doc.write (d_remaining)

* methods of Date obj —

methods	purpose.
1) getDate()	returns the date of date obj. (frm 1 to 31)
2) getDay()	(frm 0-6 : 0-sun, 1=mon..)
3) getMonth()	(frm 0-11 : 0-Jan, 1=Feb..)
4) getFullYear()	.. (4 digits)
5) getYear()	.. (2 digits : 0-99).
6) getHours()	.. (0-23)
7) getMinutes()	.. (0-59)
8) getSeconds()	.. (0-59)
9) getTime()	returns the no of msec.
10) setDate()	set date of month (frm 1-31)
11) setFullYear()	
12) setHours()	
13) setTime()	
14) toString()	converts Date obj to str.

III Array obj :

* It is a collection of values & are useful bcz you can use them to manipulate & sort grps of things.
* To create a instance of Array —
x = new Array().

* eg :-

days = new Array ("Monday, ... "Sunday")

days[2] → wednesday.

* methods of Array obj :-

methods	purpose
1). concat ()	joins 2/more arrays to create 1 new one.
2) join(separator)	joins all of the elements of an array togthr separated by the char specified as a separator (default comma).
3) reverse ()	
4) slice ()	returns specified part of A
5) sort ()	