What is Scripting?

A script is program code that doesn't need pre-processing (e.g. compiling) before being run. In the context of a Web browser, scripting usually refers to program code written in JavaScript that is executed by the browser when a page is loaded, or in response to an event triggered by the user.

Scripting can make Web pages more dynamic. For example, without reloading a new version of a page it may allow modifications to the content of that page, or allow content to be added to or sent from that page. The former has been called DHTML (Dynamic HTML), and the latter AJAX (Asynchronous JavaScript and XML).

API ➔ lib ➔ predefine programs

What scripting interfaces are available?

The most basic scripting interface developed at **W3C** is the **DOM**, the **D**ocument **O**bject **M**odel which allows programs and scripts todynamically access and updates the content, structure and style of documents. DOM specifications form the core of DHTML.

Modifications of the content using the DOM by the user and by scripts trigger events that developers can make use of to build rich user interfaces.

Types of script: Scripts are classified into the following two types.

- Client-side script
- Server-side script

Client-side script:

➔These scripts are getting executed within the web Browser (client).
➔Here we don't need any software.
➔These scripts are used for client-side validations (data verification & data validations)

Ex: JavaScript, VBScript, typescript etc…

Server-side script:

➔A script which executes in server machine with support of the web-server/app-server software's like **IIS**(Internet information services), Tomcat, JBOSS, etc.
➔ These scripts are used for server-side validations (authentication & authorization).

Ex: php, jsp, asp.net, VueScript, Express Script, nodeJS, cgi, perl etc…

What are the differences between script and language?

Script	**Language**
Weakly or loosely typed programming And lightweight	Strong or closely typed programming and HW
Easy to understand compare to PL	Complex to understand compare to Script
External libraries not required	Required
No special compiler required	Special compiler mandatory
Client side validation	Server/client side validation/verifications
Ex: JavaScript, VBScript,TypeScript, Perl, Shell etc.	Ex: C, CPP, vb.net, Java etc.

JavaScript Introduction

✓ In **1995**, JavaScript was created by a **Netscape** developer named **"Brendan Eich"**. First, it was called **Mocha**. Later, it was renamed **LiveScript**.

Mocha(1995) ➔ LiveScript ➔ JavaScript(1997-dec)

✓ **Netscape** first introduced a JavaScript interpreter in **Navigator**2.

Why is it called JavaScript?

When JavaScript was created, it initially had another name: "LiveScript". But Java was very popular at that time, so it was decided that positioning a new language as a "younger brother" of Java would help. But as it evolved, JavaScript became a fully independent language with its own specification called ECMAScript, and now it has no relation to Java at all.

ECMAScript => ES

Later JavaScript became an **ECMA**(**E**uropean **C**omputer **M**anufacturers **A**ssociation **S**cript) standard in 1997. **ECMAScript** is the official name of the language.

✓ **JavaScript** is implementation of **ES**; **ES** is the specification of JavaScript.

RBI ➔ SBI, HDFC,ICICI➔ customer
ES ➔JS ➔ Programmer

- ✓ JavaScript is originally it's not programming language.

$$\left.\begin{array}{l}\text{AJAX}\\ \text{JQuery}\\ \text{JSON}\\ \text{Angular}\\ \text{NodeJS}\\ \text{ReactJS}\\ \text{ASP.net}\end{array}\right\}\boxed{\textbf{JavaScript}}$$

These tech & frameworks using JavaScript as their PL

- ➢ JavaScript is a **Speed, light weight, Interoperability, Extended Functionality, dynamic**, **loosely typed**, **free ware** and **open-source**.

- ➢ Its single threaded programme

- ➢ **JavaScript** is an object-based or **prototype-based** programming.
  It's not **OOPS**

- ✓ JavaScript is client-side (browser-side) programming. That means it executes on the browser.
- ✓ It can also be used in server-side by using **NodeJS, ASP, JSP**
- ✓ JavaScript is a case sensitive programme (mixed case).
- ✓ To work with JavaScript, we don't need to install any software.

- ✓ JavaScript is "**<u>interpreter-based</u>**" programming, means the code will be converted into machine language line-by-line. JavaScript interpreter is already embedded in Browsers.

Parser:

JS code (high) ←→JS parser ←→ machine code

Every browser they have own JS Engine:

V8 →Chrome and Opera.

SpiderMonkey→Firefox.

Chakra→IE

SquirrelFish→ Safari

V8 → Edge

Why we Use JavaScript?

Using HTML/CSS, we can only design a web page but it's not supported to perform logical operations **such as calculations, decision making and repetitive tasks, dynamically displaying output, reading inputs from the user, and updating content on webpage at client side**. Hence to perform these entire tasks at client side we need to use JavaScript.

Where it is used?

There are so many web applications running on the web that are using JavaScript like Google, Facebook,twitter, amazon, YouTube etc.

It is used to create interactive websites.

It is mainly used for:

1. Client-side verifications and validations
2. Dynamic drop-down menus
3. Displaying date and time
4. Build forms that respond to user input without accessing a server.
5. Displaying popup windows and dialog boxes (like alert dialog box, confirm dialog box and prompt dialog box)

6. Manipulate HTML "layers" including hiding, moving, and allowing the user to drag them around a browser window.

   etc...

Limitations of JavaScript

Client-Side JavaScript has some limitations which are given below;

1. Client-side JavaScript does not allow reading and writing of files.
2. It cannot be used for networking applications.
3. It doesn't have any multithreading or multiprocessing capabilities.
4. it doesn't support db connections.

JavaScript Versions

Version	Officeal Name	Release Date
1	ECMAScript 1	June-1997
2	ECMAScript 2	June-1998
3	ECMAScript 3	Dec-1998
4	ECMAScript 4	2004 (not released)
5	ECMAScript 5	Dec-2009
5.1	ECMAScript 5.1	June-2011
6	ECMAScript	June-2015
7	ECMAScript	June-2016
8	ECMAScript	June-2017

how many ways to imp js?

JS we can develop/imp in 3 ways, but in 4place.

those are:

- inline scripting
- internal scripting
- external scripting

> inline scripting

inline script nothing but writing code within the tag, by using event/dynamic attributes

for this we need tag & event attribute

onclick, onsubmit, onfocus, oninput, onload, etc..

Syn: **<tag event="js code" event="js" event="js">**

>internal scripting

Internal script is nothing but html code and javascript code both are placed in the same file, but not in same line.

Internal script must be implemented inside **<script>** tag, <script> is a paired tag.

> scripting in head sec

head is first executed part of html, hence javascript is also executes first.

```
<head>
        <script  type="text/javascript”>
                JS code
        </script>
</head>
```

> scripting in body sec

body level script is executed after head section

```
<body>
        <script type="text/javascript”>
                JS code
        </script>
</body>
```

> external scripting

> external script is nothing but html code and javascript code designed in separate files

>type js code in sep file and save that file with "filename.js"

>re-use

>while writing external script don’t use **<script>** tag and event attribute.

Calling: **fun-name();**

External file Syn:

```
function fun-name()
{
   Steps
}
```

Fun

OR

```
{
   Steps
}
```

block

Note: external file should be saved with an extension ".js"

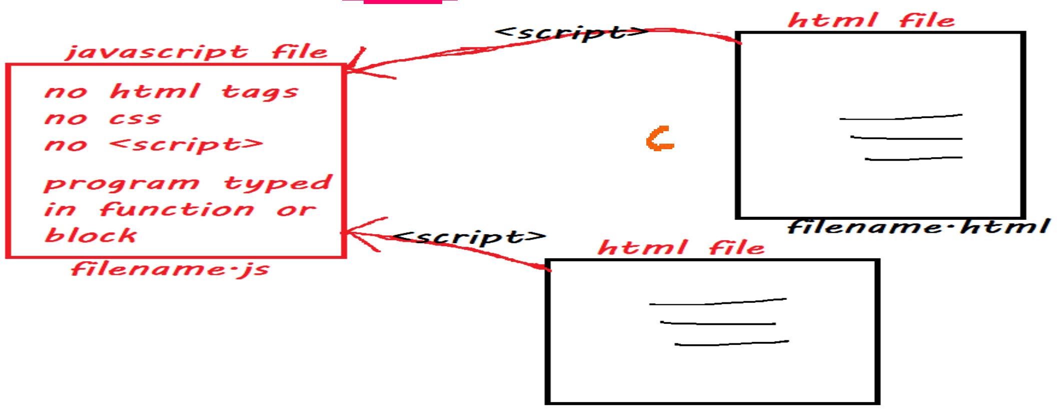
>we canaccess external script by using <script> tag from html.

> from either head nor body section

Syn:

link

```
<script src="filename1.js"></script>
<script src="filename2.js"></script>
```



Comments in JavaScript

Comment is nothing but it is a statement which is not display on browser window. It is useful to understand which code is written for what purpose.

Comments are useful in every programming language to deliver message. It is used to add information about the code, warnings or suggestions so that the end user or other developer can easily interpret the code.

Types of Comments:

There are two types of comments are in JavaScript

1. Single-line Comment ex: **//comment**
2. Multi-line Comment ex: **/* comments */**

Single-line Comment

It is represented by double forward slashes //. It can be used before any statement.

Example:

```
<script>
// It is single line comment
document.write("Hello Javascript");
</script>
```

Multi-line Comment

It can be used to add single as well as multi line comments.

It is represented by forward slash / with asterisk * then asterisk with forward slash.

Example:

```
<script>
/* It is multi line comment.
```

```
It will not be displayed */
document.write("Javascript multiline comment");
</script>
```

JS ➔ lib ➔ collection of reserve words, operators, functions, methods, classes, objects

DOM => JS’s API

- ⇨ Application Programming Interface
- ⇨ It’s an interface between JS and webpage
- ⇨ API is collection of predefine classes, and class is group properties & methods/functions
- ⇨ For working with any class we need an object
- ⇨ Dom class’s related objects are created by browser, @time of webpage loading or opening.

    window, document, console, location, cookie, history, body, head are predefine/implicit objects

object is a collection of properties & methods

window, document, console, history, navigation are predefine objects (implicit)

“**window**” is base object for all JS objects.

“window” object used for interacting with browser window to perform some operations.

“**document**” is the sub object of window.

“document” object used for interacting with web page/web document to perform some operations.

Syn: window.document or document

"**console**" is the sub object of window.

"console" object used for interacting with browser console to perform some operations.

Syn: window.console or console

Note: window, document, console, location, cookie, history, body, head are predefine/implicit objects

write() method: The write() method writes HTML expressions or javascript code to a document without line breaking.

Syn: **document**.write(val1, val2, val3….);
document.write(exp1 + exp2 + exp3...);

writeln() method: The writeln() method writes HTML expressions or javascript code to a document with line breaking.

Syn: **document**.writeln(val1, val2, val3….);
document.writeln(exp1 + exp2 + exp3...);

log() method: The log() method writes HTML expressions or javascript code on **browser's console** (press **F12** key) with line break.

Syn: console.log(val1, val2, val3…);
console.log(exp1 + exp2 + exp3...);

Example:

```
<html>
<head>
<script type='text/javascript'>
        document.write("<h1>hello world!</h1><p> have a nice day ! </p>");
</script>
</head>
```

```
</html>
```

Can we use HTML tags in write() method?
Yes, we can use tags in write().

writeln() method: The writeln() method is similar to the write method, with the addition of writing a newline character after each statement.
Syn:document.writeln(exp1,exp2,exp3 ...)

Example:

```
<!DOCTYPE html>
<html>
<body>
<pre>
        <script type='text/javascript'>
                document.writeln("Welcome to JS");
                document.writeln("Welcome to JS");
        </script>
</pre>
</head>
</html>
```

Note: You have to place writeln() in pre tag to see difference between write() and writeln().
Writeln() actually produces the output in new line (\n) but browser will not detect the \n as linebreak, hence to show it correctly and keep format as it is we will use pre tag.

Example:

```
<!DOCTYPE html>
<html>
<head>
        <script type='text/javascript'>
```

```
		document.write("<h1 style='color:blue; font-size:30px; font-family:tahoma'> Welcome To JS</h1>");
		document.write("<font color='green' size='16px' face='Arial'> Welcome To JS</font>");
	</script>
</head>
<body>
</body>
</html>
```

Note:

- the above type of code is known as DHTML
- In JavaScript a string should be in single or double quotes.
- Double quotes inside using single quotes are valid, single quotes inside using double quotes valid.

Example

```
<!DOCTYPE html>
<html>
<body>
	<script>
		document.write("JavaScript is client side script");
		document.write("<br>");
		document.write("JavScript is 'ECMA' Implementation<br>");
		document.write('JavScript released by NetScape<br>');
		document.write('NetScape release "Mocha"<br>');
	//document.write('NetScape release 'Mocha'<br>'); Error
	//document.write("NetScape release "Mocha"<br>"); Error
	</script>
</body>
</html>
```

JavaScript string with escape sequences: An escape character is consisting of backslash "/" symbol with an alphabet. The following are frequently using escape characters.

1. \n : inserts a new line
2. \t : inserts a tab space
3. \r : carriage return

4. \b : backspace
5. \f : form feed
6. \' : single quote
7. \”: double quote
8. \\ : Backslash

Difference between window.document.write&document.write:
There is no difference between these two statements,window is highest level object. It contains child objects & their methods

```
child object/sub object
        |
window.document.write();
    |       |         |
browser   page     method

document.write();
  |         |
page     method
```

browser is default object, master object, super object.
write() is a method related to document object

ex:
```
<head>
        <script type='text/javascript'>
                window.document.write("livescript is javascript");
                document.write("<br>");
                document.write('livescript is javascript');
        </script>
</head>
```

JavaScript semicolon(;):

In javascript every statements ends with semicolon(;). It is an optional notation.

ex:
```
<head>
	<script type='text/javascript'>
		document.write=("livescript is javascript");
	</script>
</head>
```

ex:
```
<head>
	<script type='text/javascript'>
		document.write("javascript");
		document.write('livescript');
		document.write('livescript is javascript');
	</script>
</head>
```

Note:
1) In the above script semicolon(;) is mandatory.
2) It is a good programming practice to use the semicolon.

JavaScript place in HTML file:
There is a flexibility given to include javascript code any where in a html document but the follow ways are most prefered in the live environment.

- script in <head>-----</head> section
- script in <body>-----</body> section
- script in <body>-----</body>&<head>-----</head> section
- script in & external file & then include in <head>-----</head> section.

JS Naming Conventions
JS => mixed case

- ⇨ Naming conventions means where we have to use uppercase and where we have to use lowercase
- ⇨ While working/using predefines items we must follow these guide lines.

class name ➔ TitleCase/Capitalilze

ex: Date, Array, NodeList, HTMLCollection

fun/method ➔ 1st word is lowercase, rest of words(2-n) are TitleCase/Capitalilze

ex: write(), log(), indexOf(), getElementById(),

variables ➔ 1st word is lowercase, rest of words(2-n) are TitleCase/Capitalilze

Ex: length, value, tagName, innerText

Constants ➔ total name in uppercase

Ex: PI, EXP, SIZE

reserved words ➔ total name in lowercase

Ex: if, else, switch, var, let, const, for, new, this, ...

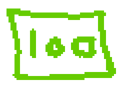
JavaScript Reserved Words:

The following are reserved words in JavaScript. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

abstract, boolean, break, byte, case, catch, char, class, const, continue, debugger, default, delete, do, double, else, enum, export, extends, false, final, finally, float, for, function, goto, if, implements, import, instanceof, int, interface, let, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, throws, transient, true, try, typeof, var, void, volatile, while, with. 59

<u>**Working with Variables**</u>

Variable is a reference name of a memory block.
Variables are created or stored in RAM(stack area).
Variables are used to store/to hold a value for reuse purpose and automatically substitute values in steps.



how to declare a variable?

we can define vars in JS Three ways, those are:

> by using "**var**"

Syn: **var** varname; ← declaration

OR

var varname=value; ← initialization

> by using "**let**" (since js6) ES6

Syn: **let** varname;

OR

let varname=value; ←init

> by using "**const**" (since js6) ES6

Syn: **const** varname=value; ← initialization

Where do we declare variables?

We can declare variables in open script tag, with in function or within block.

<u>**Rules for variable naming**</u>

- Name should start with an alphabet (a to z or A to Z), underscore(_), or dollar($) sign.

- After first character we can use digits (0 to 9).
- variables are case sensitive. for example, a and A are different variables.
- space is not allowed, means name should be single word.
- special chars (symbols) are not allowed in name, except _ and $.

for example:

var eid;	~~var 1a;~~
var total;	var a1;
var _b;	~~var book id;~~
~~var a@;~~	var studentid;
~~var #b;~~	~~var case;~~
var book_id;	var a$1

Java script did not provide any **data types** for declaring variables and a variable in javascript can store any type of value. Hence java script is loosely typed programme.

We can use a variable directly without declaring it in javascript, it’s called dynamic programming.

Var	Let	Const
We use in function or global scope	We can in function scope	We can in function scope
Block scope not supported	Block scope supports	Block scope supports
Re assigning value	Re assigning value	Not supports re assigning value
Re declaration of	Not supports	Not supports

variable supported		
Since JS1	Since JS6	Since JS6
It supports Hoisting	Not supports	Not Supports

Global Variable

var is declared with in script tag but outside function & block those are global variables.

these global variables are accessible from anywhere in program.

declared with **window** object is known as global variable.

JavaScript datatypes:

In JavaScript data types are classified into the following two cat.

1. Primitive datatypes
2. non-primitive datatypes

Primitive data types:

Primitive data types allow storing data directly. ➔ref/address
These datatypes allow us to store only 1 value @time. ➔ N vals
These are popularly known as non-reference ➔reference
SA HA
Not sharable

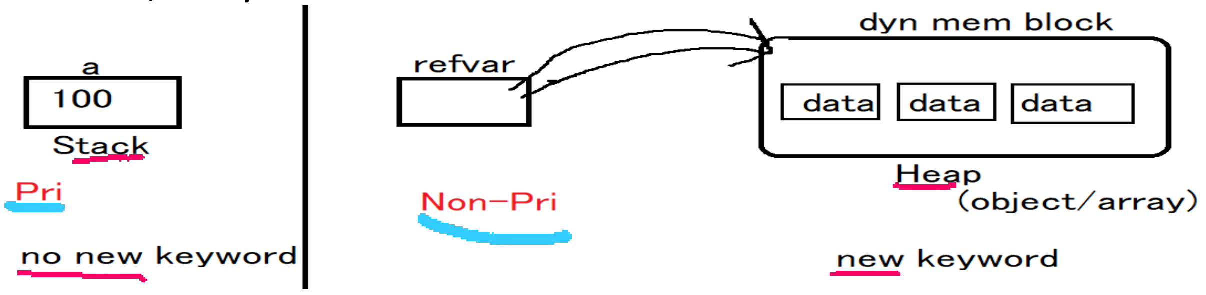
Javascript has a five primitive data types.

1. string **Ex:** "siva" 'apples123' `Dno: 1-2-3/1a`
2. number **Ex:** 10 -25 100.56 -3.7 988098009800
3. boolean **Ex:** true false
4. undefined **Ex:** value not assigned
5. null

class, array, functions

Non-primitivedatatypes: Non-primitive datatypes allow to store reference(address) of data.

These datatypes allow us to store more than 1 value @time.
These are popularly known as reference or composite data types.
Ex: class, array



var st1 = new String();

__Primitive data types__:
Strings: In javascript a String should be within a singleordouble quote.
var name="nit";
var name='nit';
Number: Javascript has only one type of numbers,they can be return with or without decimals
var x1=34.00; with decimals
var x2=34   without decimals

Boolean: It is used to represent a Boolean value,These are as follows.
var x = true //equivalent to true, yes or on
var y = false //equivalent to false, no or off

undefined: It is a value of variable with no value.
var x;   //now x is undefined

Null: variables can be emptied by setting the value to null.
ex: var x=null; //now x is null

__**typeof**__
typeof is one of reversed word, it's used to identify datatype of a variable or value.
Syn: **typeof** var-name
        typeof value

Dynamic data types: Javascript has dynamic types. This means that the same variable can be used as different types.
ex:

```
var x; //now x is undefined
var x=5; // now x is a number
var x="ram"; // now x is a String
```

<!-- example on variable declaration -->

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>JS Ex13</title>
</head>
<body>
<h1>Demo on difference between var and let</h1>
        <script>
      var a=10;  //define
document.write(a +"<br>");
           var a=20.56;   //re-defination
        document.write(a +"<br>");
      var a="apple";  //re-defination
        document.write(a +"<br>");

           let n=101;  //define
document.write(n +"<br>");
           //let n=202;  re-defination ==> Error: Identifier 'n' has already been declared
           n=202;  //changing value
document.write(n +"<br>");
        </script>
</body>
</html>
```

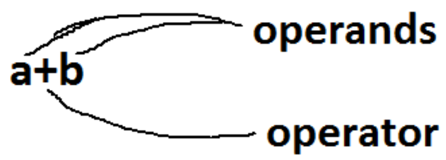
Non-primitive data types: When a variable is declared with the keyword **new**, the variable is an object.
new is used for dynamic memory allocations (for creating objectsand arrays).
these datatypes are also called as reference datatype.
Ex:

```
var st=newString();
var x=newNumber();
let y=newBoolean();
let a = [ ];
```

here **LHS** are reference variables, and **RHS** are objects.
reference variables are storing address of dynamic memory (object)

JavaScript operators

operator is a symbol (special char) and it is used to perform certain operation(task).
every operator is a symbol, but every symbol is not operator.
every operator requires some values, those are called as operands.
Ex:



Expression

Its combination of one operator and some operands

Cat:

- Unary ➔it req one operand
  - increment
  - decrement
- Binary ➔ it req two operands
  - Arithmetic
  - Relational
  - Logical

- Bitwise
- Assignment
- Concatenation

➢ Ternary ➔ it req three operands
- Conditional

Arithmetic operators: using these operators we can perform the basic math calculations.

Ope are + - * / % **

operators are:

operator	Description	example
+	addition	j+12
-	subtraction	j-22
*	multiplication	j*7
/	division	j/3
%	modulus	j%6
**	power	x**y x^y

relational operators: these operators are used to provide comparison between two operands. these are boolean operators (true/false).

Operators are: > < >= <= == != === !==

operator	Description	example
==	is equal to	j==42
!=	is not equal to	j!=17
>	is greater than	j>0
<	is less than	j<100
>=	is greater than or equal to	j>=23
<=	is less than or equal to	j<=13
===		a===b
!==		a!==b

Logical operators: these operators are used to perform multiple comparisons @time. these are boolean operators (true/false).
Operators are: && || !

operator	Description	example
&&	And	j==1 **&&** k==2
||	OR	j<100 **||** j>0
!	Not	!(j==k)

And

Cond1	**Cond2**	**Result**
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

Or

Cond1	**Cond2**	**Result**
T	**T**	**T**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

Not

Cond	**Result**
T	**F**
F	**T**

assignment operators: these operators are used to store/assign value to memory block (var/array/objects...)
Operator is =

Shorthand/compound operator is a combination of assignment and arith/bitwise.

Operators are: += -= /= *= **= &= |= >>= <<= ...

Total=total+price ➔ total+=price

operator	Description	example
=	store	a=10

shorthand:

+=	addition & assign	a+=10
-=	subtract & assign	a-=5
*=	product & assign	a*=20

/=	division & assign	a/=7
%=	modulus & assign	a%=6

Concatenation operator: this operator is used to concatenation multiple strings then formed into a single string. One operand should be string to perform concatenation. Resultant value comes in string format.
Operator is +

Bitwise operators & | ~ ^ >> <<

Ex: **"rama"+"rao" ==> "ramarao"**
"mangos"+123 ==> "mangos123"
true+"siva" ➔ "truesiva"

unary operators: these operators are used to increment or to decrement a value. operators are ++ and --
++ (increment) ==> it adding 1 to an existing value **Ex:** a++ or ++a
-- (decrement) ==> it subtracting 1 from an existing value **Ex:** a-- or --a

ternary operator: this operator is used to **decision making** operation. operator is **?:,** this operator also called as **conditional operator**.
(condition) ? statement1 : statement2

Operator Precedence Table:

The operator precedence table can help one know the precedence of an operator relative to other operators.

Precedence	Operator	Description	Associativity
1	()	Grouping	–

2	.	Member	left to right
[]	Member	left to right	obj["func"]
new	Create	–	new Date()
()	Function call	left to right	func()
3	++	Postfix increment	–
--	Postfix decrement	–	i–
4	++	Prefix increment	right to left
—	Prefix decrement	–i	
!	Logical NOT	!TRUE	
typeof	Type	typeof a	
5	**	Exponentiation	right to left
6	*	Multiplication	left to right
/	Division	18/9	
%	Remainder	4%2	
7	+	Addition	left to right
–	Subtraction	4-2	
8	<<	Left shift	left to right
>>	Right shift	y>>2	
>>>	Unsigned Right shift	y>>>2	
9	<	Less than	left to right
<=	Less than or equal	3<=4	
>	Greater than	4>3	
>=	Greater than or equal	4>=3	
in	In	"PI" in MATH	
instanceof	Instance of	A instanceof B	
10	==	Equality	left to right
!=	Inequality	x!=y	
===	Strictly equal	x===y	
!==	Strictly unequal	x!==y	
11	&	Bitwise AND	left to right
12	^	Bitwise XOR	left to right
13	|	Bitwise OR	left to right
14	&&	Logical AND	left to right
15	||	Logical OR	left to right
16	? :	Conditional	right to left
17		Assignment	right to left
+=	x+=3		
-=	x-=3		
*=	x*=3		

/=	x/=3		
%=	x%=3		
<<=	x<<=2		
>>=	x>>=2		
>>>=	x>>>=2		
&=	x&=y		
^=	x^=y		
|=	x|=y		
18	,	Comma	left to right

JavaScript dialog boxes: JavaScript has 3 kinds of dialog boxes.

1. Alert box
2. Confirm box
3. Prompt box

Alert box: An alert box is often used if you want to make sure information comes through the user. When an alert box pops up, the user will have to click "ok" to proceed.

Syn: window.alert("message"/expr);

ex:

```
<body>
        <script type='text/javascript'>
                alert("invalid entry");
        </script>
</body>
```

Note:html tags we can’t use in alert() function.

How to display multiple line on the alert:

We cann't the use <br> tag here because alert is a method of the windows object, that can’t be interpret html tag. Instead, we use the new line escape character.

```
<head>
        <script type="text/javascript">
```

```
-           alert("javascript \n is\n a\n client-side \n programming \n
language");
        </script>
</head>
```

ex: **Alert with functions**

```
<head>
        <script type='text/javascript'>
                functionmyAlert(){
                        alert("javascript \n is \n a \n client-side \n programming
\n language");
                        alert("1 \n \t 2 \n \t \t3");
                }
        </script>
</head>
<body>
        <p> click the button to display alert messages ....</p>
        <button onclick="myAlert()"> click me</button>
</body>
```

confirm box:

It is often used, if you want the user to verify and accept something. When a confirm box pops up, the user will have to click either "ok" or "cancel" to proceed. If the user clicks "**ok**" the box returns *“***true***”*. If the user clicks **"cancel"** the box returns **"false".**

Syntax: window.confirm("message");

ex:

```
<head>
        <script type='text/javascript'>
                confirm("click ok or cancel");
        </script>
</head>
```

ex:

```
<head>
```

```
	<script type='text/javascript'>
		var x=confirm("click ok or cancel");
		alert("user selected option is:"+x);
	</script>
</head>
```

ex:

```
<head>
	<script type='text/javascript'>
		var x=confirm("click ok or cancle");
		alert("user selected option is:"+x);
		if(x==true) {
			alert("user clicked on OK button");
		}
		else{
			alert("user clicked on cancel button");
		}
	</script>
</head>
```

ex: **confirm with function**

```
<head>
	<script type='text/javascript'>
		functionmyConfirm(){
			var x=confirm("click ok or cancel");
			alert("user selected option is:"+x);
			if(x==true) {
				alert("user clicked on ok button");
			}
		}
	</script>
</head>
<body>
	<p> click the button to display the user selected result..</p>
	<button onclick="myConfirm()">confirm</button>
```

```
</body>
```

Data ➔ static data

- While designing of program we are assigning values to vars
- This given by programmer
- This always same, means not changing the data execution to execution

➔ Dynamic

- While execute of program(after webpage open) assigning values to vars
- This given by user
- This always changing, means data changing the data execution to execution
- We can take the data from user, in two ways:
  - Html input elements (UI/html forms)
  - Prompt dialog

Prompt Box: It is used to, if you want the user to input a value while entering a page. When a prompt box pops up the user will have to click either "**ok**" or "**cancel**" to proceed after entering an input value. If the user clicks "ok" the box returns the **value/empty**. If the user clicks "cancel" the box returns **"null".**

Syntax: window.prompt("sometext", defaultvalue);

ex:

```
<head>
	<script type='text/javascript'>
		prompt("Enter Any Number:");
	</script>
</head>
```

ex:

```
<head>
	<script type='text/javascript'>
		varMyVal=prompt("Enter Any Number:");
```

```
                alert("User Entered value is:"+MyVal);
        </script>
</head>
```

Note: these 3methods are provided by window object.

External JavaScript with popup boxes:
step1: *Create a required js file*

```
functionMyAlert(){
        alert("welcome to externaljs");
}
functionMyConfirm(){
        confirm("click ok or cancel");
}
functionMyPrompt(){
        prompt("Enter Any Value");
}
```

Save with .js extension @ any location....!!

step2: preparing required html file.

```
<html>
        <head>
                <script type="text/javascript" src="myscript.js">
                </script>
        </head>
        <body>
                <p>Click the button to display alert message..</p>
                <button onclick="MyAlert()">Alert</button>
                <p>click the button to display confirm message...</p>
                <button onclick="MyConfirm()">confirm</button>
                <p>click the button to display prompt value..</p>
                <button onclick="MyPrompt()">prompt</button>
        </body>
```

```
</html>
```

parseInt()

predefine function => window
text based int converts into integer format

"100"	➔	100
"10.78"	➔	10
"rama"	➔	NaN (Not a Number)

Syn:

window.**parseInt("value")**

parseFloat()

predefine function => window
text based float converts into floating type

“100"	➔	100
"10.78"	➔	10.78
"rama"	➔	NaN (Not a Numeric)

Syn:

window.parseFloat("value")

Note: both are global functions

Control Statement

control statements are used to control(change) execution flow of program based on user input data.

types:

> conditional statements (dm) ➔ if
> loops (iterations) ➔ for, while, do while
> un-conditional (branching) ➔ break, continue, return
> Switch

Conditional Statements:

If Statement

The if statement is used to perform decision making operations. means if condition is true, it executes some statements. if condition is false, it executes some other statements.

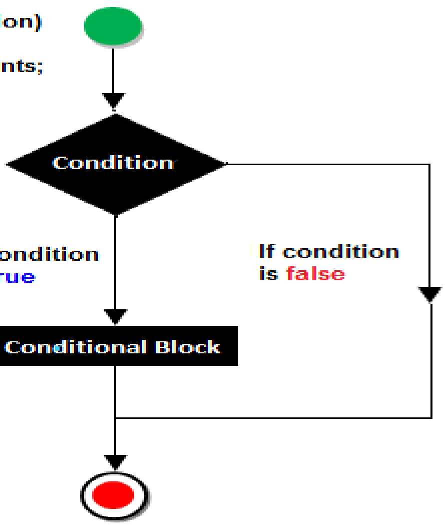
There are three forms of if statement.

- simple if
- If else
- if else if (ladder if)

If statement

if is most basic statement of Decision-making statements. It tells to program to execute a certain part of code only if particular condition or test case is true.

```
if( condition)
 {
  statements;
 }
```

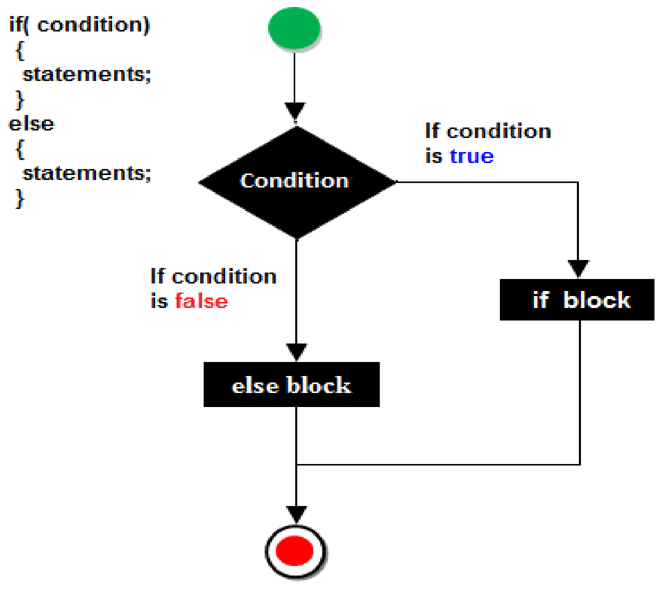


Example

```
<script>
var a=10;
if(a>5)
{
document.write("value of a is greater than 5");
}
</script>
```

<u>if-else statement</u>

In general, it can be used to execute one block of statement among two blocks.



<u>Example of ifelse statement</u>

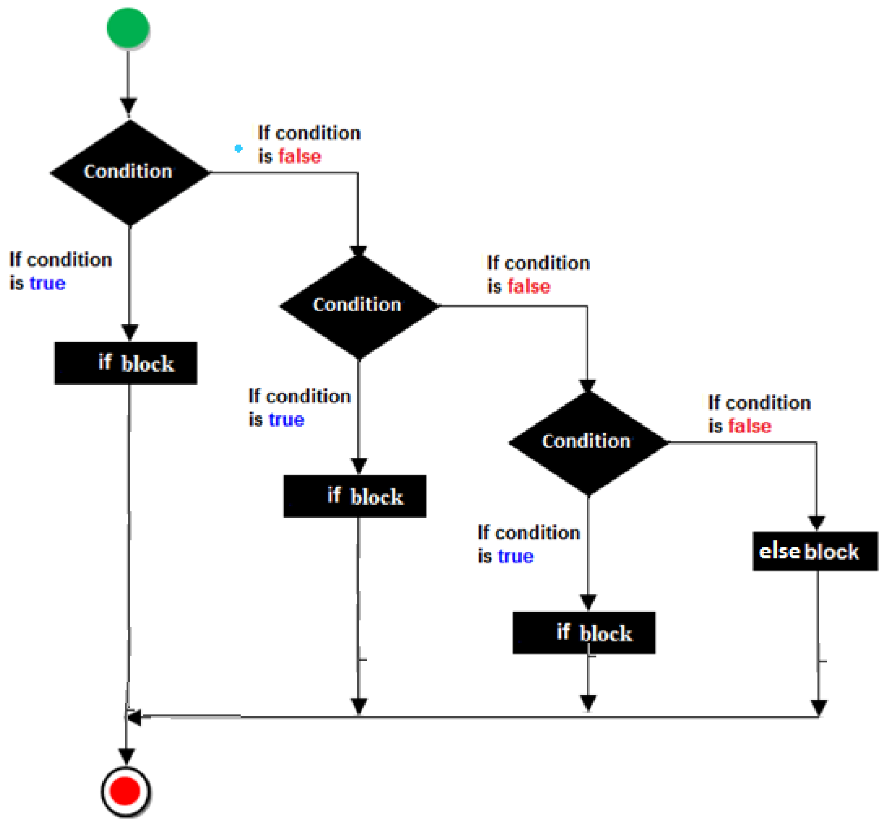
```
<script>
var a=40;
if(a%2==0)
{
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
```

Result

a is even number

JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions.



Syntax

```
if(expression1)
{
//content to be evaluated if expression1 is true
}
else
if(expression2)
{
//content to be evaluated if expression2 is true
}
else
{
//content to be evaluated if no expression is true
}
```

Example of if..else if statement

```
<script>
var a=40;
if(a==20)
{
document.write("a is equal to 20");
}
else if(a==5)
{
document.write("a is equal to 5");
}
else if(a==30)
{
document.write("a is equal to 30");
}
else
{
document.write("a is not equal to 20, 5 or 30");
}
</script>
```

switch statement

> switch is selection statement, but it's not decision making.
> its better performance.

Syn:

```
switch(var/expr)
{
        case  value:  statements...
                      break;
        case  value:  statements...
                      break;
        case ...
        default:  statements...
}
```

Looping Statement

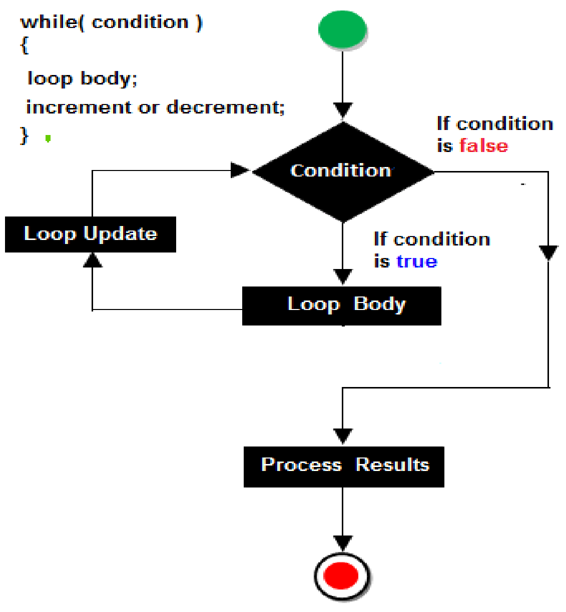
Set of instructions given to the interpreter to execute until condition becomes false is called loops. The basic purpose of loop is min code repetition.

The way of the repetition will be forming a circle that's why repetition statements are called loops. Some loops are available In JavaScript which are given below.

- while loop (top testing/entry level)
- for loop
- do-while (bottom testing/exit level)

<u>**while loop**</u>

When we are working with "while loop" always pre-checking process will be occurred. Pre-checking process means before evolution of statement block condition part will be executed. "While loop" will repeat in clock wise direction or anti-clock wise direction.



```
while( condition )
{
 loop body;
 increment or decrement;
}
```

init → starts
cond → ends
step → <u>counter</u>

body → ~~steps~~ task

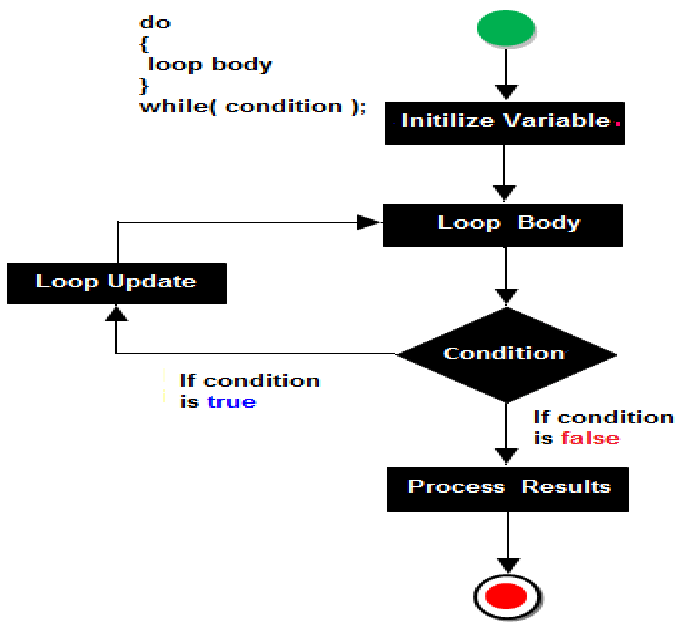
<u>Example of while loop</u>

```
<script>
var i=10;
```

```
while (i<=13)
{
document.write(i + "<br/>");
i++;
}
</script>
```

<u>do-while loop</u>

In implementation when we need to repeat the statement block at least 1 then go for do-while. In do-while loop post checking of the statement block condition part will be executed.

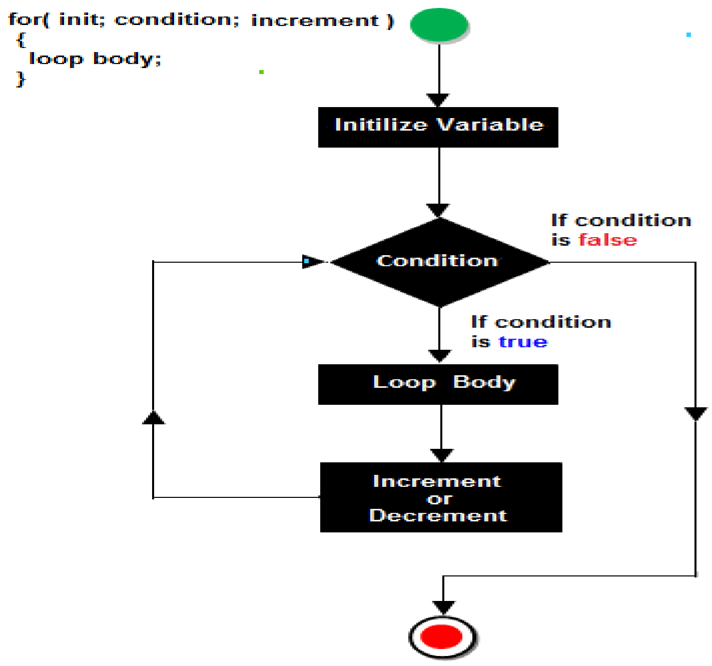


<u>Example of do-while loop</u>

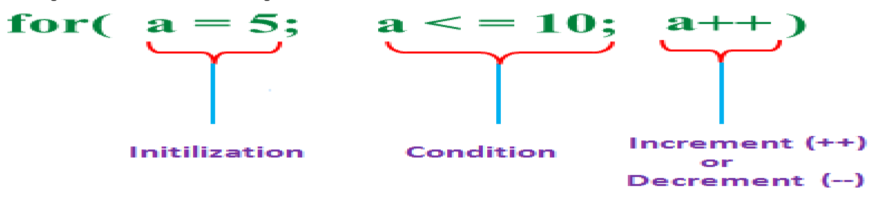
```
<script>
var i=11;
do{
document.write(i + "<br/>");
i++;
}while (i<=15);
</script>
```

for Loop

For loop is a simplest loop first we initialized the value then check condition and then increment and decrements occurred.



Steps of for loop



Example of for loop

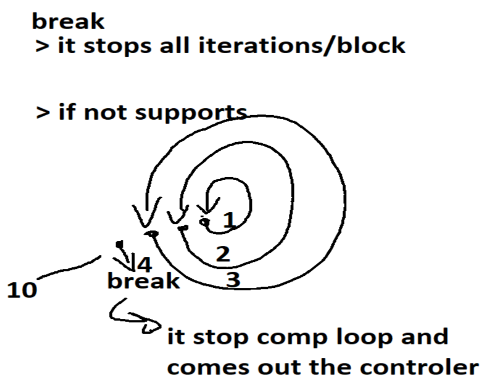
```
<script>
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
```

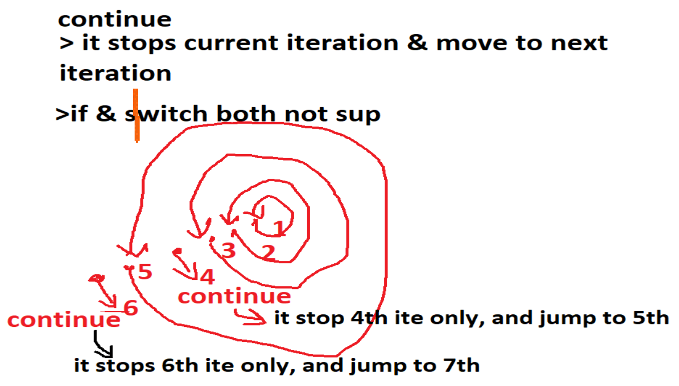
Unconditional statements

These are used to jump/skip statements execution

Types:

- break ✓
- continue ✓
- return




continue
> it stops current iteration & move to next iteration
>if & switch both not sup
1
2
3
4
continue
5
6
continue
it stop 4th ite only, and jump to 5th
it stops 6th ite only, and jump to 7th

<noscript> tag: It is used to provide an alternate contains for users when script is disabled or not supporting, It is a paired tag. It is always declared within the body section.

syntax: <noscript>------</noscript>

ex:

```
<head>
        <script type='text/javascript'>
                alert("welcome to js");
        </script>
</head>
<body>
        <noscript>
                <p    style='color:red'>oops    your    browser    not    supporting
javascript
                update/change the script settings and try..</p>
        </noscript>
</body>
```

Data, var, ope, control, arrays completed

Functions

- ➔Named funs
- ⇨ expr funs
- ⇨ arrow funs

set of instructions, are used to perform task
adv: reuse