**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.

**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”**.

**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.

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 a **Speed, light weight, Interoperability, Extended Functionality, dynamic**, **loosely typed**, **cross platform**, **free ware** and **open-source**.

**Speed** 🡺 js applications runs more faster than ...

**light weight 🡺**less codemore operations

**Interoperability 🡺**Javascript they have the capability to work within other web technologies.

**Ext Fun 🡺**

**Dynamic typed 🡺** w/o declaring vars we can directly

**loosely typed 🡺** defined any var in JS, that allows to store any type value

**cross platform 🡺** cross platform compatible

* Its single threaded programme
* **JavaScript** is an object-based or **prototype-based** programming.
* JavaScript is client-side (browser-side) programming. That means it executes on the browser.
* It can also be used in server-side by using **Node, ASP, PHP**
* JavaScript is a case sensitive programme (mixed case)
* To work with JavaScript, we don’t need to install any software.
* JavaScript is “**interpreter-based**” 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

**JS Engine:**

V8 🡺Chrome, Edge and Opera

SpiderMonkey 🡺 Firefox

Chakra 🡺IE

SquirrelFish 🡺 Safari

JS used 🡺 web dev, mobile app, gaming, animations, AI, networking app ...

**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 webpages**.

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...**

**JS6 new 🡺 ES**

**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 doesn't have any multithreading or multiprocessing capabilities.
3. it doesn't support db connections.

**How many ways to imp js?**

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

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 the same line.

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

**> scripting in head sec**

head is the first executed part of html, hence javascript is also executed 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 an external script don’t use **<script**> tag and event attribute.

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

**External file Syn:**

**function fun-name()**



**{**

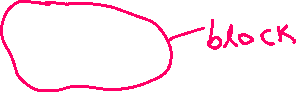
**Steps**

**}**

**OR**

**{**

**Steps**



**}**

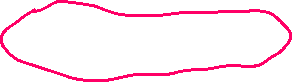
**Note: external file should be saved with an extension “.js”**

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

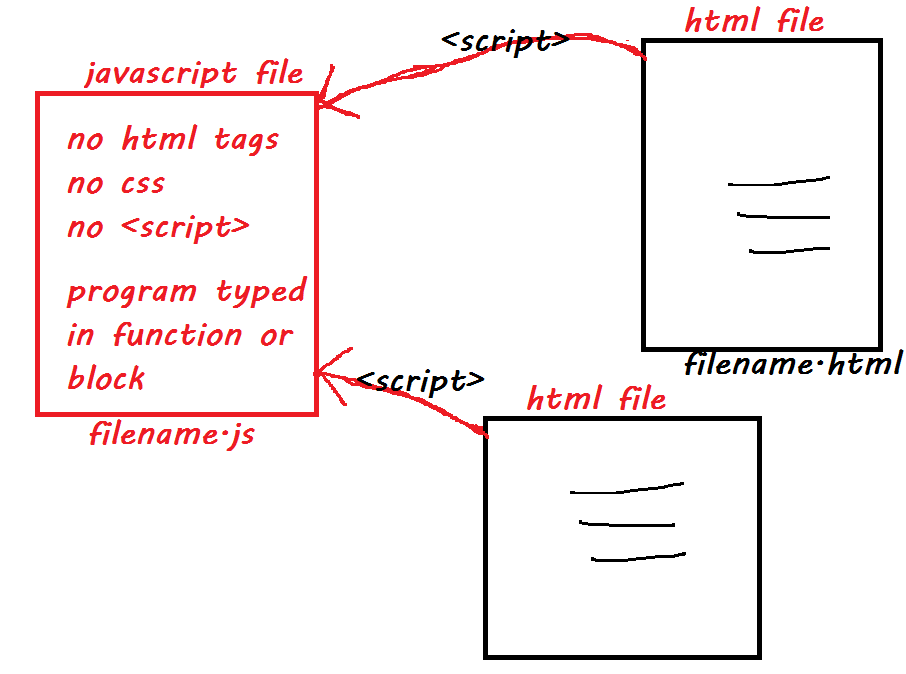
> from either head nor body section



Syn:



**<script src="filename.js"></script>**





Code in the form function/block/constructor

Function definition syn:

**function** fun-name**()**

**{**

**statements**

**}**

**Calling Syn:**

**fun-name(); 🡸 html inline or internal**

Comments in JavaScript

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

Comments are useful in every programming language to deliver messages. 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 \*/**

**JS 🡺 lib 🡺 collection of reserved words, operators, functions, methods, classes and objects (implicit)**

object is an instance of class (copy of class)

**object is a collection of properties & methods**

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

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

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

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

Syn: window.document or document

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

“console” object used for interacting with browsers console to perform some operations.

Syn: window.console or console

**Press F12 key**

**Note:** window, document, console, history etc… are predefined/implicit objects

**JS Printing methods**

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

**Syn: window.document**.write(val1, val2, val3….);

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

**Syn: window.document**.writeln(val1, val2, val3….);

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

**Syn: window.console**.log(val1, val2, val3…);

**JavaScript string with escape sequences**: An escape character consists 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

**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 predefined items we must follow these guidelines.

class name 🡺 TitleCase/Capitalize case

**ex:** Date, Array, NodeList,

HTMLCollection, HTMLHeadElement

fun/method/variable 🡺 1st word is lowercase, rest of words(2-n) are TitleCase/Capitalize

**ex:**

**vars 🡺** length, value, innerText, textAlign

**funs 🡺** alert(), prompt(), parseInt(), setInterval()

**methods 🡺 .**write() .log()

**.**getElementById()

.querySelectorAll()

constants 🡺 total name in uppercase

**Ex:** PI, EXP, SIZE

Reserved words 🡺 total name in lowercase

**Ex:** typeof,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



**Working with Variables**

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)

Syn: **let** varname;

OR

**let** varname=value;

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

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

**Where do we declare variables?**

We can declare variables in open script tag (global), within function (local) or within block (block level).

**Rules for variable naming**

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

· After the 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 a 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

**Loosely Typed**

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.

**dynamically typed**

We can use a variable directly without declaring it in javascript, it’s called dynamic typed 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 variable supported | Not supports | Not supports |
| Since JS1 | Since JS6 | Since JS6 |
| It supports Hoisting | Not supports | Not Supports |

**Global Variable**

var is declared within the script tag but outside function & block those are global variables.

These global variables are accessible from anywhere in the program.

declared with a window object is known as a 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** **Non-PDT**

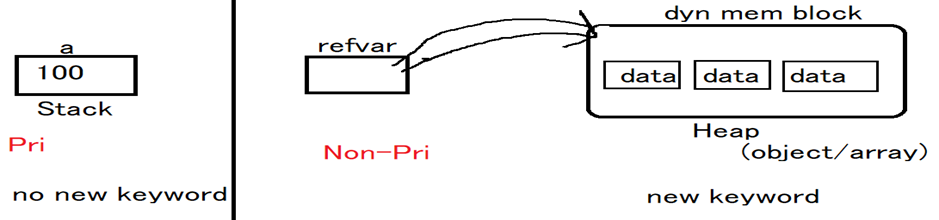
🡺PDT types allow storing data directly 🡺 reference/address

🡺PDT allow us to store only 1 value @time 🡺 N values

🡺Stack Area 🡺 Heap Area

🡺Not shareable 🡺 Shareable

🡺These are popularly known as non-reference 🡺reference data type



**Primitive data types**:

string, number, boolean, undefined, null(object)

Non-pri

Arrays, Class&object, functions

***Strings***: In javascript a String should be within a single or double 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

**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 objects and arrays).

these datatypes are also called as reference datatype.

**Ex:**

var st=**new**String();

var x=**new**Number();

let y=**new**Boolean();

let a = [ ];

here **LHS** are reference variables, and **RHS** are objects.

reference variables are storing address of dynamic memory (object)

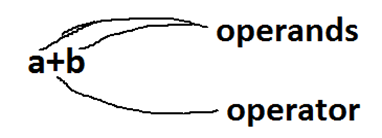
**JavaScript Operators**

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 an operator.

Every operator requires some values, those are called operands.

**Ex:**



**Expression**

Its combination of one operator and some operands

**Operator Categories**

1. Unary 🡺it requires one operand

o increment

o decrement

2. Binary 🡺 it requires two operands

o Arithmetic

o Relational

o Logical

o Assignment

o Concatenation

3. Ternary 🡺 it requires three operands

o 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 xy

***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 Or Not**

**Cond1 Cond2 Result Cond1 Cond2 Result Cond Result**

**T T T T T T T F**

**T F F T F T F T**

**F T F F T T**

**F F F F F F**

***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 è total+=price

operator Description example

= store a=10

**shorthand**:

+= addition & assigna+=10

-= subtract & assign a-=5

\*= product & assigna\*=20

/= division & assign a/=7

%= modulus & assign a%=6

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

Operator is +

**Ex: "rama"+"rao” ==> "ramarao"**

**"mangos"+123 ==> "mangos123"**

true+”siva” è “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 for decision **making** operations. operator is **?:,** this operator is also called a 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);**

**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: var = window.confirm("message");**

**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);**

**Data è static data**

o While designing of program we are assigning values to vars

o This given by programmer

o This always same, means not changing the data execution to execution

**è Dynamic**

o While execute of program(after webpage open) assigning values to vars

o This given by user

o This always changing, means data changing the data execution to execution

o We can take the data from user, in two ways:

§ Html input elements (UI/html forms)

§ Prompt dialog

**Note:**

These 3 methods do not support html tags.

**parseInt()**

predefined function => window

string based int converts into integer format

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

"100" 🡺 100

"10.78" 🡺 10

"rama" 🡺 NaN (Not a Number)

**parseFloat()**

predefined function => window

string based float converts into floating type

Syn:

**window.parseFloat("value")**

“100" è100

"10.78" è10.78

"rama" èNaN (Not a Numeric)

**Note:** both are global functions

**Parse operators**

**🡺** + is parse ope

🡺 Unary ope, use only left side (prefix)

Syn:- +”value” or +variable

+”10” 🡺 10

+”10.56” 🡺 10.56

+“ram” 🡺 NaN

**Control Statement**

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

Types:

> Conditional statements 🡺 if, Switch

> Loops (iterations) 🡺for, while, do while

**>** Unconditional (branching) 🡺 break, continue, and return

# Conditional Statements:

# If Statement

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

There are three forms of if statements.

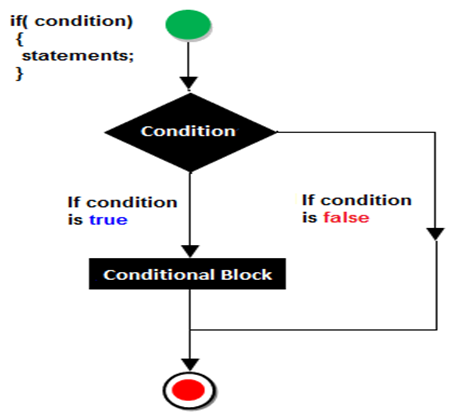
· simple if

· If else

· if else if (ladder if)

**If statement**

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



Example

<script>

var a=10;

if(a>5)

{

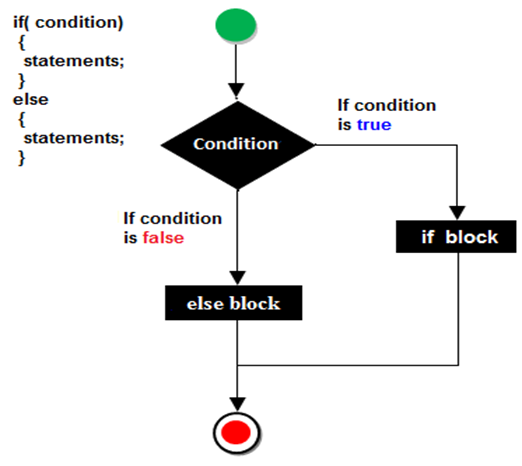
document.write("value of a is greater than 5");

}

</script>

**if-else statement**

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



Example of ifelse statement

<script>

var a=40;

if(a%2==0)

{

document.write("a is even number");

}

else{

document.write("a is odd number");

}

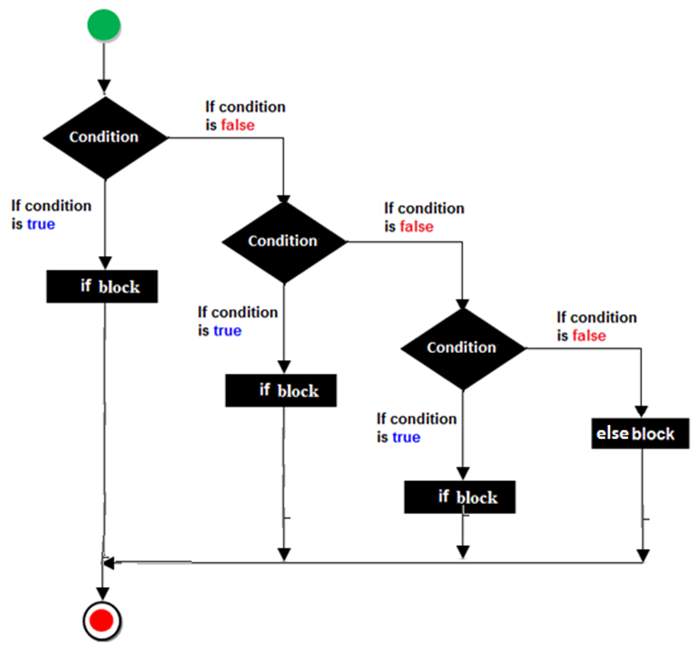
</script>

Result

a is even number

**JavaScript If...else if statement**

It evaluates the content only if the 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 a 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 Statements

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

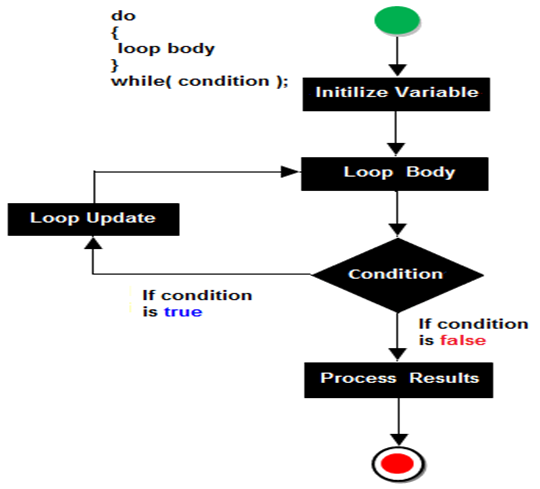
· while loop (top testing/entry level)

· for loop

· do-while (bottom testing/exit level)

**while loop**

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 clockwise direction or anticlockwise direction.



Example of while loop

<script>

var i=10;

while (i<=13)

{

document.write(i + "<br/>");

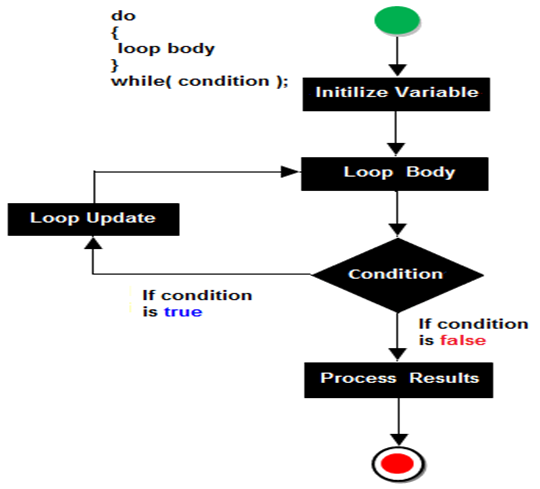
i++;

}

</script>

**do-while loop**

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



Example of do-while loop

<script>

var i=11;

do{

document.write(i + "<br/>");

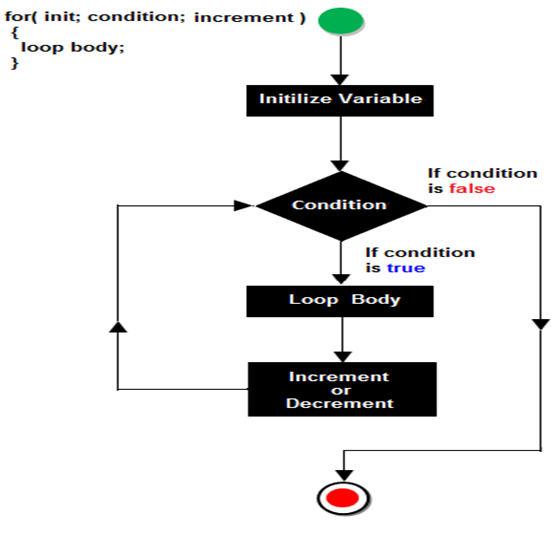
i++;

}while (i<=15);

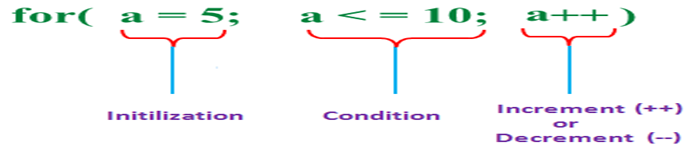
</script>

**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**

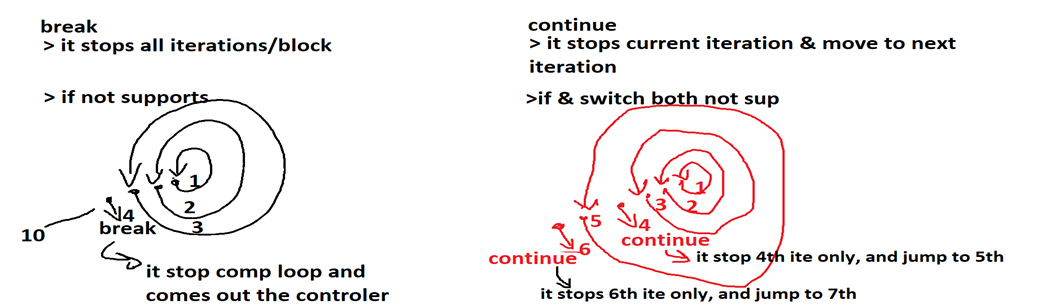
These are used to jump/skip statements execution

Types:

Øbreak

Øcontinue

Ø return

****

**<noscript> tag**: It is used to provide an alternate container for users when script is disabled or not supported, 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>