Unit - I

Basics of JavaScript Programming

12 Marks [4R, 4U, 4A]

⋆ Introduction -

JavaScript is a scripting language that enables to enhance static web applications by providing dynamic, personalized and interactive content. A script is a small piece of program that can add interactivity to the website. For example, to pop-up alert box message, or a script to provide a dropdown menu.

JavaScript, JScript, Perl, PHP, Python, VBscript are the most popular examples of scripting languages.

There are two types of scripting languages based on where do they exists or run.

1. Client-side scripting
2. Server-side scripting

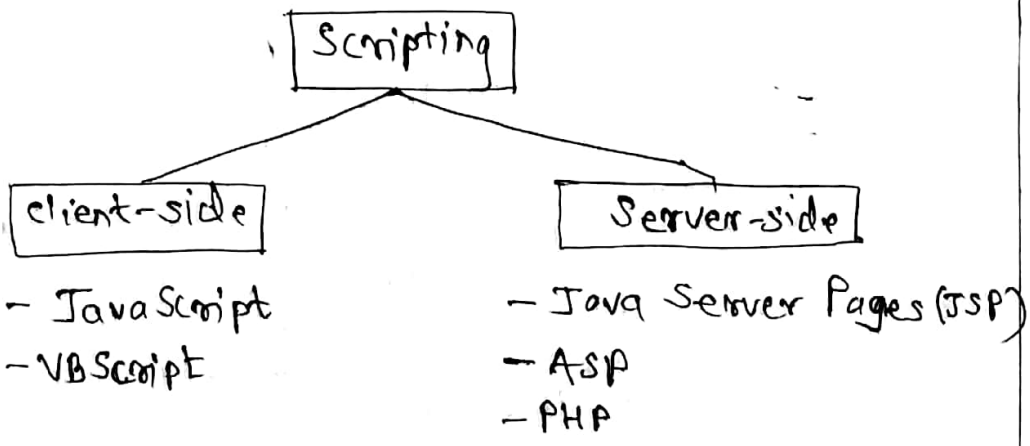


fig - Types of Scripting

Difference between client-side & server-side scripting -

① Client-side scripting -

In this type of scripting the scripts run in a browsers. The source code is transferred from the web server to user's computer over internet and run directly in the browser.

It is useful for making pages more interesting and user friendly. It can also provide gadgets such as calculator, clock etc.

It needs to be enabled on the client computer.

Example are - JavaScript, HTML, CSS

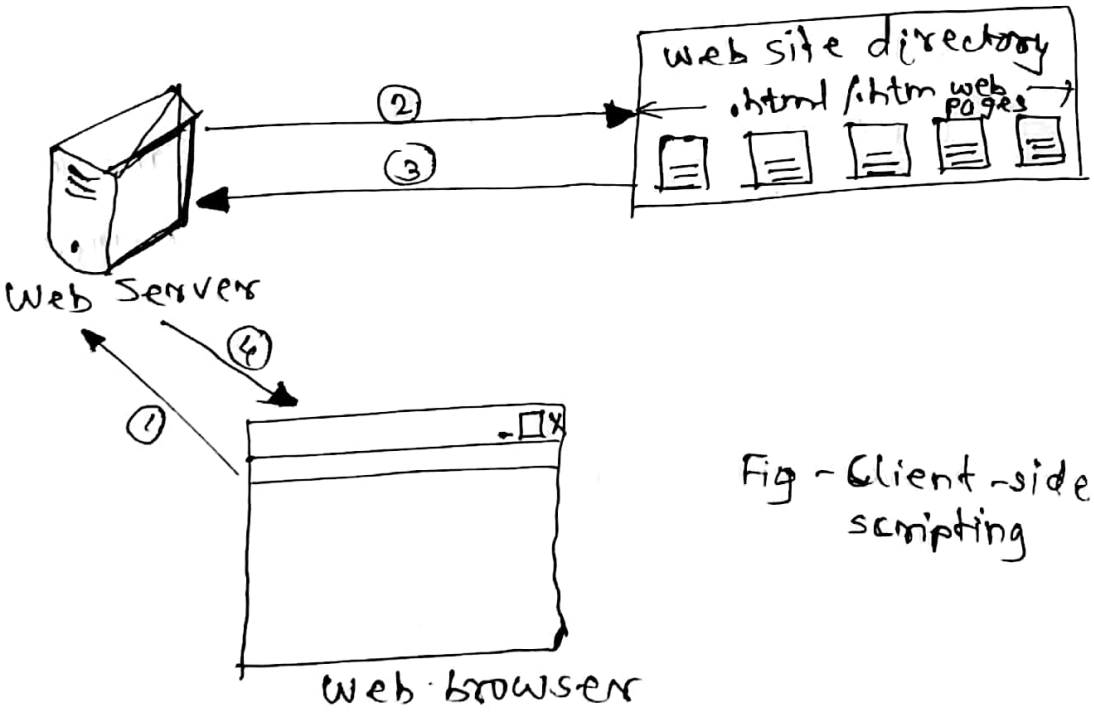


Fig - Client-side scripting

The process with client-side scripting is as follows:

step ① - The web browser sends a request for a particular web page of web site to hosted web server.

step ② - The web server searches for page in web site directory.

step ③ - On finding requested page, the web server collects HTML of web page.

step ④ - The web server sends the HTML to browser which the displays the web page.

Advantages -

1) Allows more interactivity
2) Execute quickly
3) Can give developers more control over look and behavior of their web widgets.
4) May improve the usability of web sites

Disadvantages of client-side -

1) Not all browsers support scripts
2) Different browsers & browser versions supports scripts differently.
3) More development time and effort required.

② Server-side scripting -

In this type of scripting the scripts run on a web server. The server contains the webpages and other contents. The server sends pages to the user/client on request. The process of server-side is as follows:

Step ① - The web browser sends a request for a particular web page of a web site to web server hosting that site.

Step ② - The web server searches for page in web site directory.

Step ③ - On finding the page, web server collects the content (code + HTML) of web page.

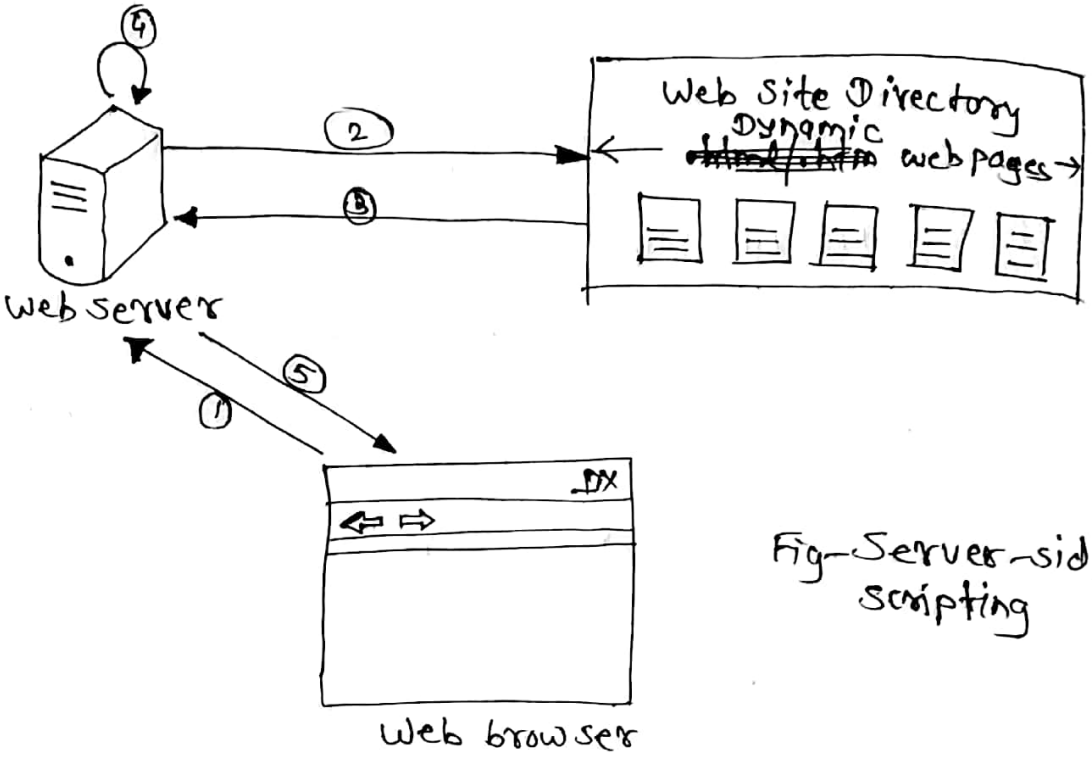


Fig- Server-side scripting

Step (4)- The web server then parses the content to produce HTML.

Step (5) - The HTML stream is sent back to the requesting browser which then parses it to develop the visual presentation of the page.

Some examples of Server-side scripting are Java Servlets, Java Server Pages (JSP), PHP, ASP.

Advantages -

1) More secure
2) Runs faster

Disadvantages -

1) Requires scripting s/w installed on server m/c.
2) Users have to wait as every time page is submitted to server.

(*) Introduction to JavaScript - (JS)

JavaScript is an interpreted, client-side, event-based, object-oriented scripting language that is use to add dynamic interactivity to web pages. JavaScript was developed by Brendan Eich at Netscape in September 1995. JS is a scripting language primarily used on the web.
JS initially name was LiveScript, Sun Microsystems later joined the Netscape & then they developed LiveScript & later its name is change to JavaScript.
JS programs run by an interpreter built into the user's web browser (not on the server).
It also name as ECMAScript

1.1 Features of JavaScript -

1) Light-weight, interpreted programming language.
2) Scripting language and it is not Java.
3) Browser support
4) JS is case sensitive.
5) Object based language. as it provides predefined objects.
6) Giving the user more control over the browser.
7) Interpreter based scripting language.
8) Most of control statement syntax same as C language.

Advantages of JavaScript –

1) Speed – Is runs fast as code runs on browser.
2) Simplicity – Simple to learn and implement
3) Versatility – can be used in a huge variety of applications.
4) Server load – Reduces website server load.
5) Less server interaction – Saver server traffic.
6) Easy to debug – easy to debug & testing
7) Richer interfaces – drop-down menu, sliders, etc.
8) Immediate feedback to ~~users~~ visitors – User don't have to wait for a page ~~reload~~ reload.
9) Increased interactivity – create interfaces that react when user mouse over them.
10) Lightweight and Interpreted – allows to build interactivity to static HTML pages.
11) Build Dynamic web pages
12) Display alert boxes
13) Validate information in forms

Disadvantages of JavaScript –

1) Security – user's computer can expoited for malicious ~~programs~~ purpose.
2) Relying on end user – As Is sometimes interpreted differently by different browsers.
3) Does not allow the reading or writing of files
4) Can not be used for Networking applications.
5) Does not have any multitasking or multi-proces capabilities.

Applications of JavaScript –

JavaScript is mainly used in following

1) Websites
2) Web Applications
3) Presentations
4) Server Applications
5) Games, etc.

Softwares required for running Javascript programs –

You can use softwares like Notepad and Browser which installed already in the system. Except this you can also use following IDE or softwares

1) Notepad ++
2) Sublime
3) Atom
4) Visual Studio Code (VS Code)
5) Microsoft FrontPage
6) Macromedia Dreamweaver, etc.
7) Webstorm
8) Netbeans

How to run first JavaScript program -

The steps to run your first JavaScript program are given below:

step ① - Open "Notepad" and create an HTML document with Head and body sections.

```
<html>
<head>


</head>
<body> ... </body>
</html>
```

step ② - In HEAD section insert the javascript block as given below. <SCRIPT> tag

```
<script>
    document.alert("Hello world!");
</script>
```

step ③ - Save this file as Sample.html on folder.

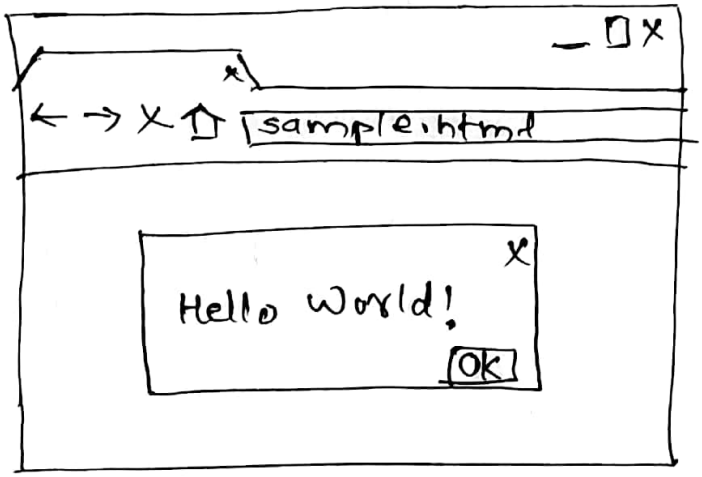
step ④ - Now to run this file double click on it then in browser you will get the output

Program and its output is given in next page.

HTML Code – sample.html

```
<html>
<head>
    <script>
        document.alert("Hello World!");
    </script>
</head>
<body>
</body>
</html>
```

Output –



JavaScript Syntax –

JavaScript can be implemented using <script> .... </script> HTML tags in a webpage.

You can place <script> tags anywhere within a web page but it is recommended to put it in HEAD section.

```
<script>
    JavaScript code
</script>
```

The <SCRIPT> tag takes two important attributes
① Language – specifies what scripting language you are using
② Type – specifies type and value which should be "text/javascript"

```
<script language="javascript"
        type="text/javascript">
    Javascript code
</script>
```

Comments in JavaScript –

JavaScript supports both C-style and C++-style comments.

① Single line comment → //.....
② Multiline comment /*.....*/
③ HTML comment <!-- .... -->
④ Javascript comment <!--

//-->

⋆ Ways to embed Javascript in HTML –

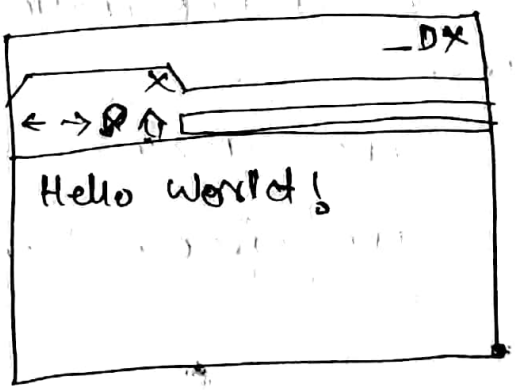
1. In the HEAD section
2. In the BODY section
3. In both HEAD and BODY section
4. In External JavaScript (.js) file

Example – Write a JavaScript program to display "Hello World!" message.

```
<html>
<head>
    <title> Hello World Example </title>
    <script>
        document.write("Hello World!");
    </script>
</head>
<body>

</body>
</html>
```

output



(1.2) Object Name, Property, Method, Dot syntax, main event –

JavaScript is an object-based language so everything is an object in JavaScript. An object is a thing such as a document, a computer, a pencil or a car.

✱ Object Name –

A typical web page contains many objects. for example a web page has two forms which are uniquely different. based on fields, buttons, etc. Each object must be uniquely identified by a name or ID. Forms could be defined form1 & form2. we can objects in Javascript

① Using constructor of object –

The new operator is used to create an instance of object

```
var obj = new object();          // Generic object
var dob = new Date(1995,4,10);   // Date object
var mycar = new Car();           // User-defined object
```

② Using Literal Notation –

A object literal consists of comma seperated list. of name-value pairs wrapped in curley braces.

```
var user = { };                  // An empty object
var circle = {x:10, y:21, radius:4};
```

⊛ Property –

Property is the value that is associated with an object. Objects can have many values depending on type of object.

For example – ① a form object has a title, a color, a width and a height.

② A window has background color, width & height.

Each object has its own properties.

We can access the properties of an object using two ways

① object-name.property; user.fname;

② object-name["property"]; user["fname"];

// Program to demonstrate object & its properties

```
<html>
 <head>
    <title> object Property </title>
 </head>
 <body>
    <script>
       var user = { fname : "Rahul",
                    lname : "Tamkhane" };
       document.write ("User = " + user.fname
                       + " " + user.lname);
    </script>
 </body>
</html>
```

output – User = Rahul Tamkhane

⊛ Methods –

Method is basically an action performed by an object. An object is like a noun and method is like as a verb.
A method is a sequence of actions (statements) performed by an object when it receives a message. for example – a submit button on login form will submit the form details to server-side application.

```
<script>
    var user = {
        fname: "Rahul",
        lname: "Tamkhane",
        fullName: function () {
                return this.fname + " " + this.
                                          lname;
            }
    };
    document.write ("User=" + user.fullName);
</script>
```

output –

User= Rahul Tamkhane

In JavaScript methods are created by using a function keyword.

```
function ()
{
    ==
    ==
}
```

⊛ Dot syntax -

An object is associated with properties (kind of information) and methods (kind of behaviours)

For example, document object has a property background color bgColor and a method write.

Then we can access the properties and methods by using the dot syntax along with object name and its property or method.

For example -

```
document.bgColor
document.write()
```

⊛ The main Event -

An event is something that causes JavaScript to start executing the code. For example when you click on some button using mouse then corresponding action will be performed.

An event is a specific circumstance that is monitored by JavaScript and that script can respond to it in some way.

some examples of events are:

1) A document loading
2) A user clicking a mouse button and
3) The browser screen changing size.

An event handler which is a part of JavaScript that reacts to important events.

For example, an even handler on fliptart website will enlarge image when user mouse over the product.

Build-in objects in JavaScript –

JavaScript has built-in objects. In browsers there is an object named window. It is called as global object. Some of built-in objects are given below.

① window object – top-level object represents a window or a frame

Properties – ActiveXObject, closed, innerHeight, innerWidth, length, screenX, screenY.

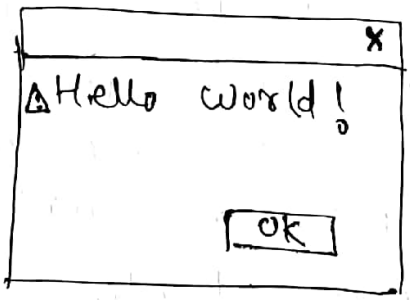
Methods – ~~① alert() method~~

① alert() – It displays a box containing a message.

Syntax → `window.alert(message);`

Example → `window.alert("Hello World!");`
or `alert("Hello World!");`

Output →



② confirm() – It displays a confirmation dialog that contains an optional message as OK & Cancel buttons.

Syntax → `var retval = window.confirm(msg);`

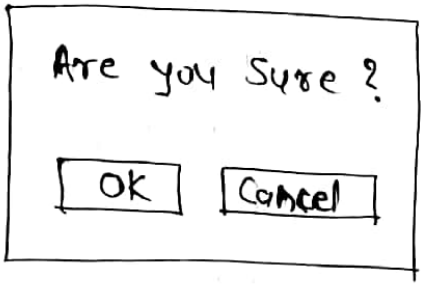
where, msg is string message
Returns true if click ok
else return false on Cancel

window.confirm(-||-); OR

Example → var r = confirm("Are you sure?");

```
if (r==true)
    document.write("ok");
else
    document.write("Cancel");
```

Output →



③ prompt() – It displays a box that prompts the user with a message and an input field.

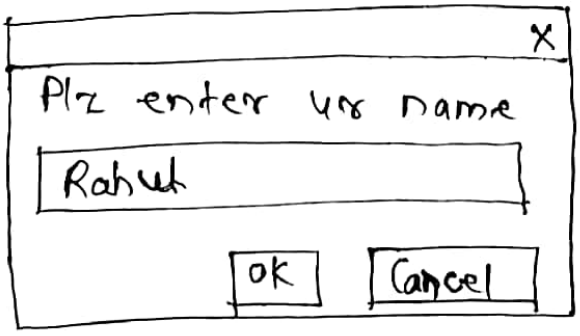
Syntax → var val = window.prompt(msg, defstr);

where msg is the message string and defstr is the default value of input field

Example →

```
var name = window.prompt("Plz enter ur
                name", "Rahul");
```

output →



② Math object - Provides properties & methods for mathematical constants and functions

Syntax → Math.PI
Math.sin(30)

Properties → E, LN2, LN10, PI, SQRT2, etc.

Methods → abs(), acos(), asin(), ceil(), cos(), sin(), floor(), exp(), max(), min(), pow(), round(), sqrt(), etc.

③ Date object - It is used to work with date and time.

Methods → getDate(), getDay(), getMonth(), getFullYear(), getHours(), getMinutes(), getSeconds(), etc.

Example →
```
var d=new Date();
document.write(d.getDay());
```

④ Java Script Identifiers

⑤ Java Script Keywords

(1.3) Values and Variables -

✱ Values -

In HTML, all values are treated as text. Means if you have enter 10, then it is not consider as number 10 it is a text 10.

JavaScript uses six kinds of values

1. Number - A numeric value that can be used in calculation
2. String - A text enclosed within quotation marks.
3. Boolean - A value that is either false or true, also represented as zero / non-zero.
4. Null - Means nothing. Null is the absense of any value.
5. Objects - An object is a value. For example, document, window, form objects.
6. Functions - A function performs an action when you call the function in JS. such as alert() function is used to display message.

We can also use literals to represent values in JS. These are fixed values. Different types of literals are

① Array literals – A list of zero or more expressions, each represents an array element, enclosed in square brackets ([ ]).
Eg → var cars = ['Bmw', 'Audi'];

② Boolean literals – Has two values true or false

③ Integer literals – Expressed in decimal (base 10), hexadecimal (base 16), octal (base 8) & binary (base 2).

Decimal (0-9) → without leading 0
Eg. 117, 458, -347

Octal (0-7) → leading 0
Eg. 015, 08, -077

Hexadecimal (0-9, A-F, a-f) → leading 0x
Eg. 0x112, 0x0011

Binary (0 and 1) → leading 0b (or 0B)
Eg. 0b11, 0B10.

④ Floating-point literals – Have decimal integer, decimal point (.), fraction & exponent.
Eg → 17.5, 6.02E21.

⑤ String literals – Zero or more characters enclosed in double (") or single (') quotes.
Eg → 'Rahul', "123", etc.

~~Values~~

⊛ Variables –

Variables are used to store some values. They are the "containers" for storing information. A variable's value can change during the script.

~~The~~

Declaring a variable –

The variable name can consists of any letter, digit and an underscore, but it cannot begin with a digit.

To declare a variable in JS use `var` keyword, which tells the browser that next follow will be the variable name.

Syntax → `var variable-name;`

Example → `var color;`

Initializing a variable –

To initialize a variable place a value in a variable using assignment operator. (=)

Syntax → `variable-name = value;`

Example → `color = "Red";`

// Program to demonstrate variables

```
<html>
<body>
  <script>
      var a=15;
      document.write("Value of a is"+a);
  </script>
</body>
</html>
```

(1.4) Operators and Expressions –

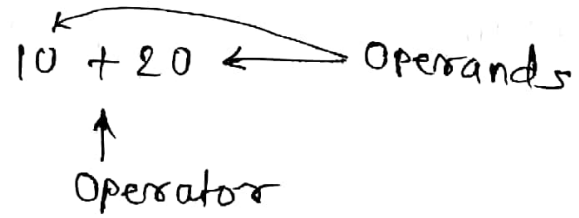
JavaScript statement is used to tell the browser to do something. It contain a mathematical expression that tells the browser to perform a mathematical operation.

A mathematical expression consists of two parts:

① Operand – An operand is the value (numbers)

② Operator – An operator is the symbol that tells the browser how to evaluate expression.

For example a mathematical expression



An operator is a special symbol indicates a certain action or operation when used with one or more operands.

JavaScript language supports variety of operators :

① Arithmetic operators —

Arithmetic operators are used to perform arithmetic operations like addition, subtraction, multiplication, etc. let us A = 10 & B = 20 then

Operators	Example
+ (Addition)	A+B = 30
− (Subtraction)	A−B = −10
* (Multiplication)	A*B = 200
/ (Division)	B/A = 2
% (Modulus)	A%B = 1
++ (Increment)	A++ = 11
−− (Decrement)	A−− = 9

② Comparision Operators — (Relational operators)

These operators are used in logical statement to determine equality or difference between variables or values. It is used to compare two values. They return boolean values true or false.

< (less than)	<= (less than or equal to)
> (greater than)	>= (greater than or equal to)
== (equal to)	!= (not equal to)

③ Logical operators —

Logical operators are used to combine two logical expressions into one expression. They are used in decision making. It evaluates expression to either true or false.

&& (Logical AND)

|| (Logical OR)

! (Logical NOT)

④ Bitwise operators —

Bitwise operators perform bit-level operations.

& (Bitwise AND)

| (Bitwise OR)

^ (Bitwise XOR)

~ (Bitwise NOT)

<< (Bitwise Left shift)

>> (Bitwise Right shift)

>>> (Bitwise shift Right with zero)

⑤ Assignment operators – (Shorthand operators)

Assignment operators are used to assign values to varibles. They are also known as shorthand operators.

= += -= *= /= %=

A = A + 5 will be written as A += 5

A = A * 10 —"— A *= 10

⑥ Conditional operators – (Ternary operator) (? :)

The conditional operators also known as ternary operator, tells the browser to take a specific action after evaluating expression.

Expression ? Value1 : Value 2

If expression is true then value 1 is use otherwise value 2 is use

Eg → 10 > 5 ? 10 : 5

⑦ Order of operations

⑧ Expressions –

An expression is a phrase wed to evaluate to produce a value. when one or more operators are combined with one or more operands then it is called an expression. Eg. (x+y)*z.

① Primary Expressions –

These are simplest expressions which are standalone, they do not include any simpler expressions. In JS these are constants or literal values, keywords & variable references.

```
1.23        // A number literal
"hello"     // A string literal
/pattern/   // A regular expression literal
```

true, false, null, this } keywords

i, sum } variables

② Object and Array Initializers –

whose value is a newly created object or array. Sometimes called as "object literals" and "array literals".

An array initializer is a comma-seperated list of expressions contained within square brackets. ([ ])

```
[ ]                    // An empty array
[1+2, 3+4]             // A 2-element array
var matrix = [[1,2,3], [4,5,6], [7,8,9]];
                          ↳ Nested array
```

object initializer is a comma-seperated list of expressions contained within curly brackets ({ }). Each subexpression is prefixed with a property name & a colon :

```
var p = { x: 2.3, y: -1.2 };   // An object with 2 properties
var q = { };                   // Empty object with no properties
q.x = 2.3 ; q.y = -1.2;        // q has same property as p
var rectangle = { upperLeft: { x: 2, y: 2 },
                  lowerRight: { x: 4, y: 5 } };
```

③ Function Definition Expressions -

It defines a JavaScript function & the value of such an expression is the newly defined function. Also known as "function literal".

A function definition expression consists of :

(a) function keyword followed by a comma-seperated list of zero or more parameters in parenthesis and

(b) a block of JavaScript code in curley braces.

```
var square = function (x) { return x*x; }
```

This function return the square of the value passed to it.

④ Property Access Expressions –

It evaluates the value of an object property or an array element.

JS defines two syntaxes for property access:

```
expression . identifier
expression [expression]
```

Eg →

```
var o = {x: 1, y: {z: 3}};
var a = [o, 4, [5, 6]];
o.x                o.y.z
o["x"]             a[1]
a[0].x             a[2]["1"]
```

⑤ Invocation Expressions –

An invocation expression in JS is syntax for calling a function or method. It starts with a function expression that identifies the function to be called.

Eg →

```
f(o)
Math.max(x, y, z)
arr.sort()
```

⑧ JavaScript Statements – [Condition statement]

A condition statement is a type of JS statement that tells the browser to evaluate a condition and based upon this evaluation, either execute or skip one or more statements.

The three types of condition statements are

1. if statement
2. switch...case statement and
3. loop statement

1.5 If statement –

The if statement is most powerful statement used for decision making.

The if statement has three parts

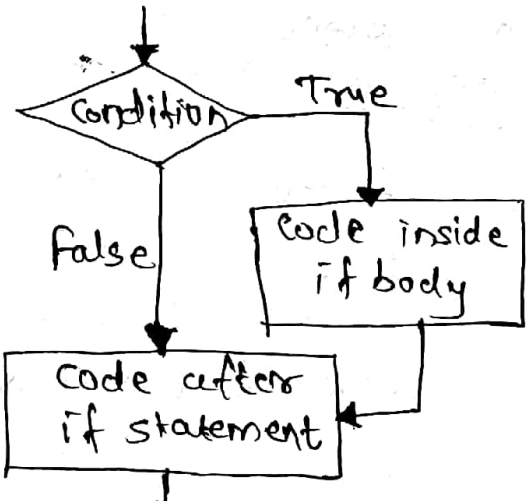
1. the if keyword
2. a conditional expression and
3. the code block that contain statements that are executed if expression is true.

Syntax

```
if (condition)
{
    // code here

}
```

Flowchart



If condition is true then code in curly braces get executed else not.

⊛ if...else statement –

Enhanced version of simple if statement. The if...else statement tells the browser "if the condition is true, then execute these statements, else execute other statements.

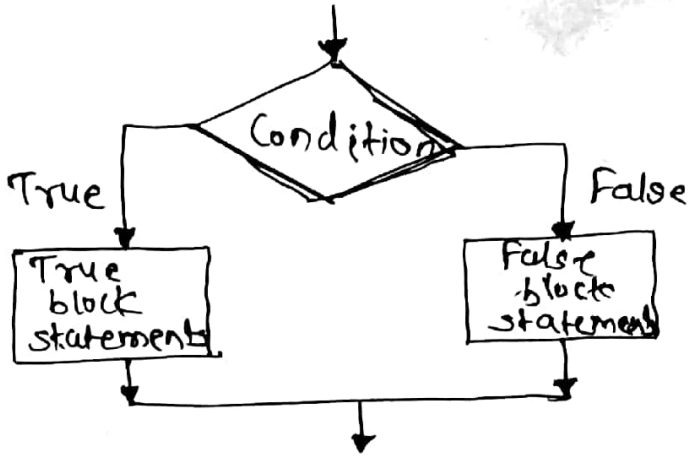
Syntax

```
if (condition)
{
    //True block

}
else
{
    //False block

}
```

Flowchart



⊛ if...else if statement –

This is just like else_if ladder statement in C. If one condition is false then it will check of second and so on, at last there is a default statement.

Syntax

```
if (condition 1)
{
    ==
}
else if (condition2)
{
    ==
}
else
{
    ==
}
```

⟹ if...else - if...else statement

Syntax

```
if (cond 1)
{
    ==
}
else if(cond2)
{
    ==
}
else
{
    ==
}
```

⊛ Nested if statement –

we can also make nesting of if statements. If we use one if...else statement within the body of another if...else statement then it is called as nested – if statement.

Syntax

```
if (condition 1)
{
    if (condition 2)
    {
        ==
    }
    else
    {
        ==
    }
}
else
{
    ==
}
```

1.6 Switch...case statement –

A switch...case statement tells the browser to compare a switch value with a series of switch cases. If a switch value matches a case value, then the browser executes statements that are placed in the case value.

A switch...case statement might has eight parts:

① The switch keyword

② A switch value compared to case values must be within parenthesis.

③ The case keyword

④ A case value is compared to switch value must be placed between the case keyword and a colon.

⑤ Case statements, executes if case value matches

⑥ The break keyword (optional), to skip all other cases

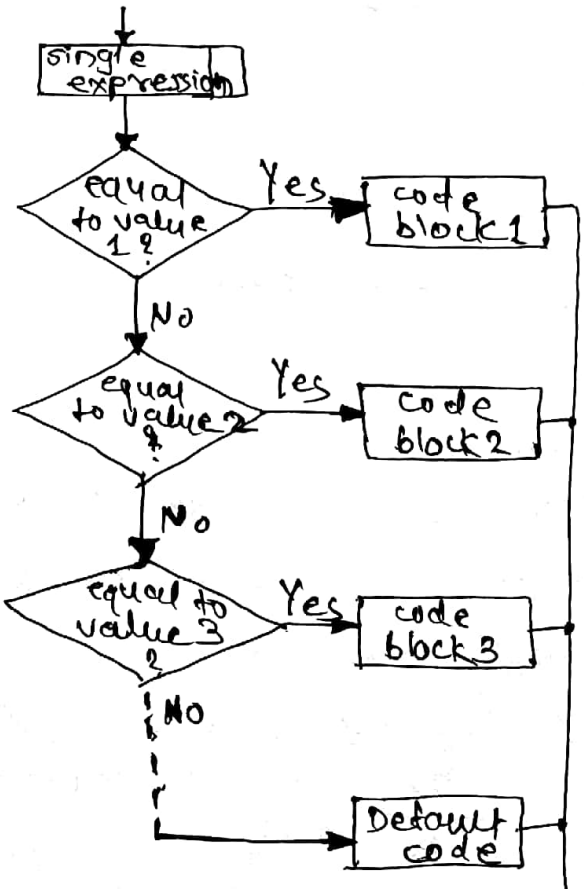
⑦ The default keyword (optional) contains statements that are executed if non of case values match the switch value.

⑧ Open and end curly braces define the body of switch...case.

Syntax

```
switch (expr/value)
{
  case value1:
    ===
    break;
  case value2:
    ===
    break;
  default:
    ===
}
```

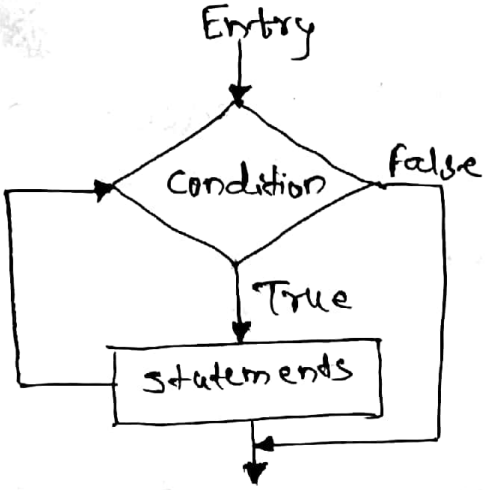
flowchart



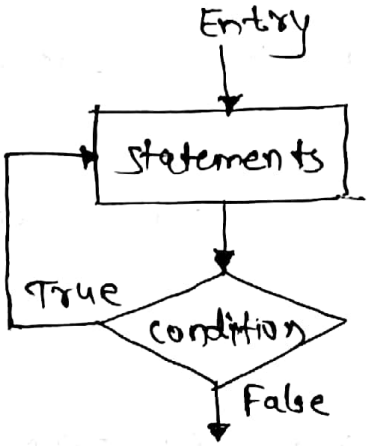
(1.7) Loop statement -

A loop is used to execute one or more statements repeatedly. Looping is the process that repeats a single or a block of statements in a program until a specified condition become true. There are two types of loops

1. Entry-controlled loop
2. Exit-controlled loop



(a) Entry-controlled loop



(b) Exit-controlled loop

JavaScript has four types of loops :

1. for loop
2. for...in loop
3. while loop
4. do...while loop

① The for loop –

The for loop tells the browser to execute statements within the loop until a condition statement return. false. The for loop has three parts:

1. initialization
2. condition
3. increment / decrement

Syntax

```
for ( initialization ; condition ; increment )
{
    // code statements here.
}
```

// Program to display numbers from 1 to 10.

```
<html>
<head>
    <title> Display nos. 1 to 10 </title>
</head>
<body>
    <script type="text/javascript">
        for (var i=1 ; i<=10; i++)
        {
            document.write(i);
            document.write("<br/>");
        }
    </script>
</body>
</html>
```

Output

1
2
3
4
5
6
7
8
9
10

② The for in loop –

The for in loop is a special kind of for loop that is used whenever you don't know the number of times that the browser should loop. It is basically used to retrieve all the properties of an object; when we don't know how many properties an object have.

The for in loop has four parts:

① The for keyword
② The list, which is placed between parenthesis
③ Open and close curly braces that define code block
④ statements that are placed within code block

Syntax

```
for (list)
{
   //statements;
}
```

⇒

```
for (variableName in Object)
{
   //statements;
}
```

```
// Program to display all properties of document object.
<html>
<body>
  <script type="text/javascript">
     var a;
     document.write("Document object properties");
     for (a in document)
     {
        document.write (a).
        document.write ("<br />");
     }
  </script>
</body>
</html>
```

③ while loop –

The while loop tells the browser to execute one or more statements continually as long as a condition is true. while loop is a type of entry-controlled loop means first condition is checked if it is true then the statement in while block gets executed otherwise nothing is executed in the block. This loop executes zero or more times.

Syntax –

```
while (condition)
{
    //statements;
}
```

// Program to display table of 2 using while loop

```
<html>
<head>
    <title> While loop </title>
</head>
<body>
    <script>
     <!--
        var i= 1;
        while (i<=10)
        { document.write (i*2);
          document.write ('<br>');
          i++;
        }
    //-->
    </script>
</body>
</html>
```

Output

2
4
6
8
10
12
14
16
18
20

Mr. Rahul S. Tamkhane

(4) The do while loop –

The do...while loop is same as while loop, except that statements within the code block executes at least once. It is a type of exit-controlled loop means first statements are executed after that condition is checked. The do..,while loop executes one or more times.

Syntax –

```
do
{
    //statements;
}while (condition);
```

// Program to display sum of numbers from 1 to 10.

```
<html>
<head>
    <title> Sum of 1 to 10 numbers </title>
</head>
<body>
    <script>
    <!--
        var i=1, sum=0;
        do
        {
            sum = sum + i;
            i++;
        }while (i<=10);
        document.write ("sum =" +sum);
    //-->
    </script>
</body>
</html>
```

⊛ continue statement – Loop control statement

The continue keyword instructs the browser to stop executing statements within the loop immediately and return to top of the loop to reevaluate the conditional expression.

Syntax – continue;

For example if we want to display numbers 1,2,3 and 5 we don't want to display 4 then continue keyword is used.

```
<script>
    var i=0;
    while (i<=5)
    {   i++;
        if (i==4)
        {
            continue;
        }
        document.write(i);
    }
</script>
```

⊛ break statement – Loop control statement

The break statement is used to break the loop. It is used to jump out of a loop or switch.

Syntax – break;

(1.8) Querying and Setting Properties –

To obtain the value of a property, use the dot (.) or square brackets ([]) operators.

① If using dot operator, the right-hand side must be identifier

```
var author= book.author;
var name = author.surname;
```

//Get "author" property of book
//Get "surname" property of author

② If using square brackets, the value within brackets must be a property name.

```
var title= book["main title"];
var author = book["author"];
```

//Get "main title" property of book

To create or set a property use a dot or square brackets but put them on on left-hand side of an assignment expression.

```
book.edition =5;
book["main title"] ="ECMAScript";
```

//Create an "edition" property of book
//set "main title" property.

(★) Deleting Properties –

The delete operator removes a property from an object.

```
delete book.author;
delete book["main title"];
```

//The book object now has no author property

The delete operator only deletes own properties not inherited ones.
After deleting property if you query the same property then it will return "undefined".

⭐ Property Getters and Setters —

An object propery is a name, a value and a set of attributes. The value may be replaced by one or two methods known as a getter and a setter.
Properties defined by getters and setters are also called as accessor properties to distinguish them from data properties that have a simple value.

(1) When a program queries the value of an accessor property then JavaScript invokes the getter method (passing no parameters).

(2) When a program sets the value of an accessor property then JavaScript invokes the setter method, passing the value of right-hand side of the assignment.

Note — a) If property has both getter and setter method then it is read/write property.
b) If property has only a getter method then it is read-only property
c) If property has only a setter method then it is write-only property.

Accessor properties are represented by "getter" and "setter". In an object literal they are denoted by get and set.

```
let obj = {
    get propName() {
        //getter
    },
    set propName(value) {
        //setter
    }
};
```

Example -

```
<html>
<head>
  <title> Getters & Setters </title>
<head>
<body>
<script type="text/javascript">
    var myCar = {
        /* Data properties */
        defName : "Toyota",
        defColor : "red",

        /* Accessor properties (getters) */
        get name() {
            return this.defName;
        },
        get color() {
            return this.defColor;
        },
```

```
        /* Accessor Properties (setters) */
        set name(newName) {
            this.defName = newName;
        },
        set color(newColor) {
            this.defColor = newColor;
        }
    };
    document.write("Car name:" + myCar.name +
                   "Car color :" + myCar.color);
    /* calling the setter accessor properties */
    myCar.name = "Audi";
    myCar.color = "blue";

    /* checking the new values */
    document.write("Car name:" + myCar.name);
    document.write("Car color:" + myCar.color);
  </script>
</body>
</html>
```
