JAVA SCRIPT

JavaScript is a very powerful client-side Scripting language.
JavaScript is used mainly for enhancing the interaction of a user with the webpage

WHY JAVASCRIPT

- Java Script offers lots of flexibility.
- with Javascript you can find tons of frameworks and libraries already developed, which can be used directly in web development.

USES OF JAVASCRIPT

Web Applications.
Mobile Applications.
Web-based Games.
Back-end web Development.

FEATURES OF JAVASCRIPT

Case Sensitive
Control Statement
in-built Functions.
Looping Statement.
Client Side Technology.

APPLICATION OF JAVASCRIPT

Javascript is used to create interactive websites. It is mainly used for:-

- Client-side validation
- Dynamic drop-down menus,
- Displaying date and time.
- Displaying pop-up windows and dialog boxes (confirm box, and Prompt dialog box and alert box)
- Displaying clocks etc.

JAVASCRIPT SYNTAX

```
<Script>
        document.write("Hello JavaScript");
</script>
```

The **Script** tag specifies that we are using JavaScript.
The document.write() function is used to display dynamic content through Javascript. We will learn about document object in detail later.

PLACES TO PUT JAVASCRIPT CODE

There are three places to put javascript code :-

- Between the body tag of html.
- Between the head tag of html.

] Inpage Js.

- In .js file. (external java Script)

1. Code between the body tag.

```
<html>
<body>
    <Script>
        alert("Hello Javascript");
    </Script>
</body>
</html>
```

2. Code Between the head tag.

```
<html>
  <head>
    <Script>
      alert("Hello Javascript");
    </Script>
  </head>
  <body>
  </body>
</html>
```

3. External JavaScript file

We can create external JS file and embed it in many html Page.

External JS file

```
alert("Hello this is External JS file");
```

Save this file .JS Extension.

html file (index.html)

```
<html>
  <head>
<script type="text/Javascript" Src="message.JS">
  </script>
</head>
  <body>
    <p> Welcome to JavaScript </p>
</body>
</html>
```

JAVASCRIPT COMMENTS

The JavaScript comments are meaningful way to deliver message. It is used to add information about the code, warnings or suggestions so that end user can easily interpret the code.

The JavaScript Comment is ignored by the JS engine i.e. embedded in the browser.

TYPES OF JAVASCRIPT COMMENTS

There are two types of comments in JS:-

- Single-line Comments.
- Multi-line Comments.

1. JAVASCRIPT SINGLE LINE COMMENTS

It is represented by double forward Slashes (//). It can be used before and after the Statement.

```
<html>
<body>
<Script>
// Single line Comment ignored by browser.
document.write("Hello JavaScript");
</script>
</body>
</html>
```

JAVASCRIPT MULTILINE COMMENTS

It can be used to add single as well as multiline Comments.
It is represented by forward slash with astorik with forward slash.

```
/* Your CODE Here */
```

```
<html>
  <body>
  <script>
/* It is a multiline Comment
   It will not be displayed */
   document.write("example of javascript multiline
      Comment");
</script>
</body>
</html>
```

JAVASCRIPT VARIABLE

A JavaScript variable is simply a name of storage location.

There are two types of variables in javascript.

- Local variables
- Global variables.

There are some rules while declaring a javascript variable (also known as identifier)

Name starts with a letter (a to z or A to Z), underscore (_), or dollar ($) sign.

After first letter we can use digits (0 to 9), for example value 1.

Javascript variable are case sensitive, for example x and X are different variables.

Example :-

```
<html>
    <body>
<script>
    var x = 10;
    let y = 20;
    Const z = x+y;

    document.write (z);
</script>
</body>
</html>
```

JAVASCRIPT LOCAL VARIABLE

A JavaScript local variable is declared inside block or function. It is accessible within the function or block only.

For example :-

```
<script>
    function abc()
{
    var x = 10; // local variable
}
</script>
```

JAVASCRIPT GLOBAL VARIABLE

A JS global variable is accessible from any function. A variable i.e. declared outside the function or declared with window object is Kfa global variable.

For example :-

```
<script> var data = 200; // global variable
    function a() {

        document.writeln(data);
    }
a(); // calling js function
</script>
```

JAVA SCRIPT DATA TYPES

JS Provides different data types to hold different types of values.. There are two types :-

- Primitive Datatype
- Non-primitive Datatype (composite)

PRIMITIVE DATA TYPES

The Primitive Data types are the lowest level of the data value in JS. There are five types of Primitive Datatypes:-

String	Represents Sequence of character e.g. "hello".
Number	Represents numeric value. eg. 100.
Boolean	Represent Boolean Value either false or true.
Undefined	Represent undefined value.
Null	Represent null ie no value at all.

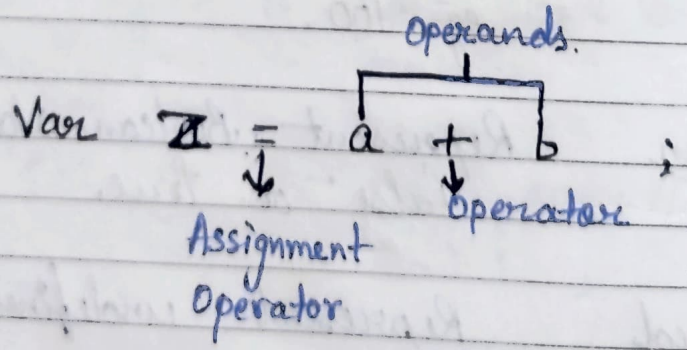
NON-PRIMITIVE DATA TYPES

The non-primitive data types contain some kind of structure with primitive data.

Object	Represents instance through which we can access member.
Array	Represents group of similar values.
RegExp	Represents regular expression.

JAVASCRIPT OPERATOR

An Operator performs some operation on single or multiple operands (data value) and produces a result.



Javascript includes following categories of Operators.

- Arithmetic Operator.
- Comparision Operator.
- Logical Operator.
- Assignment Operator.
- Conditional Operator.

ARITHMETIC OPERATOR

It is used to perform mathematical Operation b/w numeric Operands.

+	Adds two numeric Operands.
-	Subtract right Operand from left Operand.
*	Multiply two numeric Operands.
/	Divide left Operand by right Operand.
%	Modulus Operator. Returns remainder of two Operands.
++	Increment Operator. Increase Operand value by one.
--	Decrement Operator. Decrease value by one.

COMPARISION OPERATOR

It is Provides Comparision Operators that compare two Operands and return a boolean value true or false.

==	Compares the equality of two Operands without considering type.
===	Compares equality of two Operands with type.
!=	Compares inequality of two Operands.
>	Returns a boolean value true if the left-side value is greater than the right-side value; otherwise, return false.

< Return a boolean value true if the left-side value is greater than or equal to the right-side value; otherwise, returns false.

>= Returns a boolean value true if the left-side value is greater than or equal to the right-side value; other return false.

<= Returns a boolean value true if the left-side value is less than or equal to the right-side value; otherwise, returns false

LOGICAL OPERATORS

In JavaScript, the logical operators are used to combine two or more conditions. JavaScript provides the following logical operators :-

&& (logical AND) both operands are true, then the condition become true.

&& is k/a AND operator. It checks whether two operands are non-zero or not (0, false, undefined, null or "" are considered as zero). It returns 1 if they are non-zero; otherwise, returns 0.

|| (logical OR) if any of the two operands are true, then the conditions become true.

|| is k/a OR operator. It check whether any one of the two operands is non-zero or not

(0, false, undefined, null or "" is considered as zero). It returns 1 if any one of them is non-zero; otherwise, returns 0.

! (Logical not) Result opposite to the operands value.

! is k/a NOT Operator. It reverses the boolean result of the operand (or condition)
!false returns true and !true return false.

ASSINGMENT OPERATORS

JavaScript provides the assignment operators to assign values to variables with less key strokes.

=	Assigns right operand value to the left operand.
+=	Sum up left and right operands values and assigns the result to the left operands.
-=	Subtract right operand values from the left operand value and assigns the result to the left operand
*=	Multiply left and right operands values and assigns the result to the left operands.

/= Divide left operands value by right operand value and assign the result to the left operand.

%= Get modulus of left operand divided by right operand and assign resulting modulus to the left operands.

CONDITIONAL OPERATOR (Ternary OPERATOR)

JavaScript provides a special operator called Ternary operator :? that assigns a value to a variable based on some condition. It is the only javascript operator that takes three operands. a condition followed by a question mark (?)

Syntax :-

```
<Condition> ? <value1> : <value2>;
```

:- Important Points :-

- Javascript includes operators that perform some operation on single or multiple operands and produce a result.
- It is the short form of if-else condition.

CONDITIONAL STATEMENTS.

JavaScript includes if/else conditional statements to control the program flow, similar to other programming languages.

JS includes following forms of if-else statements :-

(i) if statement.
(ii) if-else statement.
(iii) else if statement.

-:- if STATEMENT -:-

Use if statement if you want to execute something based on some condition.

Syntax :-

```
if (boolean expression)
{
    // code to be executed if condition
       is true.
}
```

-!- else CONDITION -!-

- use else statement when you want to execute the code every time when if Condition evaluates to false.
- The else statement must follow if or else if Statement. Multiple else block is NOT allowed.

Syntax :-

```
if (condition expression)
{
    // Execute this code.
}
else {
    // Execute this code.
}
```

-!- else if CONDITION -!-

use "else if" condition when you want to apply second level condition after if statement.

Syntax :-

```
if ( condition expression )
{
    //Execute this code block.
}
else if ( condition expression ) {
// Execute this code block.
}
```

points to be Remember.

- Use if - else condition statement to control the Program flow.
- JavaScript includes three forms of if condition: if-else, if condition, if-else if condition.
- The if confirm condition must have conditional expression in brackets () followed by single statement or code block wrapped with {}.
- 'else if' statement must be placed after if condition. It can be used multiple times.
- Else condition must be placed only once at the end. It must come after if or else if statement.

JAVA SCRIPT SWITCH

The Switch is a conditional statement like if statement. Switch is useful when you want to execute one of the multiple code blocks based on the return value of a specified expression.

Syntax :

```
Switch (expression or literal value){
  Case 1:
    // code to executed
  break;
  Case 2:
    // code to be executed
  break;
  Case n:
    // code to be executed
  break;
  default:
    // default code to be executed
    // if none of the above case executed
}
```

Points to be Remember :

- The Switch is a conditional statement like if statement.
- A Switch statement includes literal value or is expression based.
- A break Keyword is used to stop the execution of case block.

JAVA SCRIPT LOOPS

The JavaScript loops are used to iterate the piece of code using for, while, do-while or for-in loops. It makes the codes compact. It is mostly used in Array.

There are four types of loops in JS :-

- For Loop
- While Loop
- do-while Loop
- for-in loop

JAVASCRIPT FOR LOOP

The JavaScript for loop iterates the elements for the fixed number of times. It should be used if number of iteration is known.

Syntax :-

```
for (initialization ; Condition ; increment)
{
    code to be executed
}
```

JAVASCRIPT WHILE LOOP

The JavaScript while loop iterates the Elements for the infinite number of times. It should be used if number of iteration is not known. The syntax of while loop is given below.

Syntax;

```
while (condition)
{
    Code to be executed
}
```

JAVASCRIPT DO-WHILE LOOP

The JavaScript do while loop iterates the elements for the infinite number of times like while loop.

Syntax :

```
do {
    Code to be executed
} while (condition);
```

JAVASCRIPT For In Loop

The JavaScript for in statement loops through the properties of an object:

Syntax :

```
for (key in object)
{
// code block to be executed.
}
```

JAVASCRIPT ARRAY

Arrays are basically collections of some items, you can write many names of the fruits in a single array.
We can store multiple value under one name.

Array are variable which can hold more than one value,

```
Var fruits = ["banana", "apple", "grapes"]
Var a = [1, "Name", false]
```

You can be store different data types value.

Array Element →	10	20	30	40	50	60
Index →	0	1	2	3	4	5

Lower bound

upper bound

Assign Value

```
Var a = [1, 2, 3, 4, 5];
document.write (a);
document.write (a[0]);
```

```
let numbers = [1, 7, 9, 21]
              0  1  2  3
numbers[0] - 1
numbers[1] - 7
```

Finding the length

```
let number = [1, 7, 9, 21]
number.length → 4
```

.length print the No. of element in a array.

Changing the Value

```
let number = [7,2,40,9]
number[2] = 8
```

"Number" now become [7,2,8,9] Arrays are mutable. array be changed.

In JS arrays are objects. The type of operator on arrays returns objects.

Java Script array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in Javascript

- By array literal
- By creating instance of Array directly. (using new keyword).
- By using an Array constructor (using new keyword).

JAVASCRIPT ARRAY LITERAL

```
Var arrayname = [value1, value2, .... valueN];
```

Values are contained inside [] and Separated by, (comma) and end of colon (;).

Example:-

```
<Script>
    var fruits = ["Apple", "Banana", "Pineapple"];

    for (i=0; i<fruits.length; i++) {
      document.write(fruits[i] + "<br/>");
    }
</script>
```

The .length property returns the length of an array.

Output:-

Apple

Banana

Pineapple

JAVASCRIPT ARRAY DIRECTLY (NEW KEYWORD)

```
Var arrayname = new Array();
```

Here, new Keyword is used to create instance of Array.

Example :-

```
<Script>

var i;

var Adit = new Array();

Adit [0] = "Anshika";
Adit [1] = "KumKum";

for (i=0; i< Adit.length; i++) {

document.write(Adit[i] + "<br>");

}
</script>
```

Output :-

Anshika

KumKum.

JAVASCRIPT ARRAY CONSTRUCTOR (NEW KEYWORD)

var arrayname = new Array (value1, value2, ...valueN);

Example :-

```
<Script>

    var emp = new Array ("Anshika", "KumKum");

    for (i=0; i< emp.length; i++)
{
    document.write(emp[i]+"<br>");
}
</script>
```

Output :-

Anshika

KumKum

JavaScript Array Methods

Concat() — It returns a new array object that contains two or more merged array.

copyWithin() — It copies the given part of the array with its own elements and returns the modified array.

entries() — It creates an iterator object and a loop that iterates over each key/value pair.

every() — It determines whether all the elements of an array are satisfying the provided function conditions.

flat() — It creates a new array carrying sub-array elements concatenated recursively till the specified depth.

indexOf() — It searches the specified element in the given array and returns the index of the first match.

isArray() — It tests if the passed value is an array.

join() - It join the elements of an array as a string.

keys() It creates an iterator object that contains only the keys of the array then loops through these keys.

map() It calls the specified function for every array element and returns the new array.

pop() It removes and returns the last element of an array.

Push() It adds one or more element to the end an array.

reverse() It reverses the elements to the of given array.

shift() It removes and returns the first element of an array.

unshift() It adds one or more elements in the beginning of the given array.

Sort() It returns the element of the given array in a sorted order.

Splice () It add / remove elements to / from the given array.

Slice () It returns a new array containing the copy of the part of the given array.

toString () It converts the elements of a specified array into array into string from, without affecting the original array.

values () It creates a new iterator object carrying values for each index in the array.

Of () It creates a new array from a variable number of arguments, holding any type of arguments.

find () It returns the value of the first elements in the given array that satisfies the specified condition.

forEach () It invokes the provided function once for each element of an array.

JAVASCRIPT OBJECT

A JavaScript object is an entity having State and behaviour (properties and Method). Javascript is an object-based language. There are 3 ways to create objects :-

- By literal
- By creating instance of object directly (using new keyword)
- By using an object constructor (using new keyword).

BY OBJECT LITERAL

object = { property1 : value1, property2 : value2 ... N }

'value' separated by colon (:)

Example :-

```
<script>

emp = {id: 102, name :"Anshika", Salary : 40000}

document.write (emp.id + " " + emp.name + " " +
                emp.salary);

</script>
```

Output :- 102 Anshika 40000

BY CREATING INSTANCE OF OBJECT

```
var objectname = new Object ();
```

Example :-

```
<Script>

Var emp = new object ();

    emp.id = 101;
    emp.name = "Anshika";
    emp.Salary = 50000;

document.write ( emp.id + " " + emp.name + " " +
                 emp.Salary );

</script>
```

Output :-

101 Anshika 50000

BY USING OBJECT CONSTRUCTOR

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.
This keyword refers to the current object.

```
function functionname (col1, col2, col3) {
    this.col1 = col1;
    this.col2 = col2;
    this.col3 = col3;
}
```

Example :-

```
<script>
    function emp(id,name, salary) {
    this.id = id ;
    this.name = name ;
    this.salary = salary ;
    }
    e = new emp(103, 'Anshika', 3000);
document.write(e.id + e.name + e.salary);
</script>
```

Output :-

103 Anshika 3000

The various methods of objects are as follows:

Object.create() This method is used to create a new object with specified prototype object and properties.

Object.is() This method determines whether two values are the same value.

Object.values() This method returns an array of values.

Object.keys() This method returns an array of a given object's own property names.

JAVASCRIPT FUNCTION

A function is a block of code that performs a specific task.

It takes in input, processes it, and Produces output. Function always returns the value.

DECLARING FUNCTION

Syntax :-

```
function nameoffunction () {

    // function body

}
```

Example :-

```
// declaring a function named greet()
function greet () {

    Consde.log ("Hello there");

}
```

CALLING FUNCTION

Syntax :-

```
// Call function
function name ();
```

```
function greet () {
  // Code
}

greet ();
// Code
```

function call

Example :-

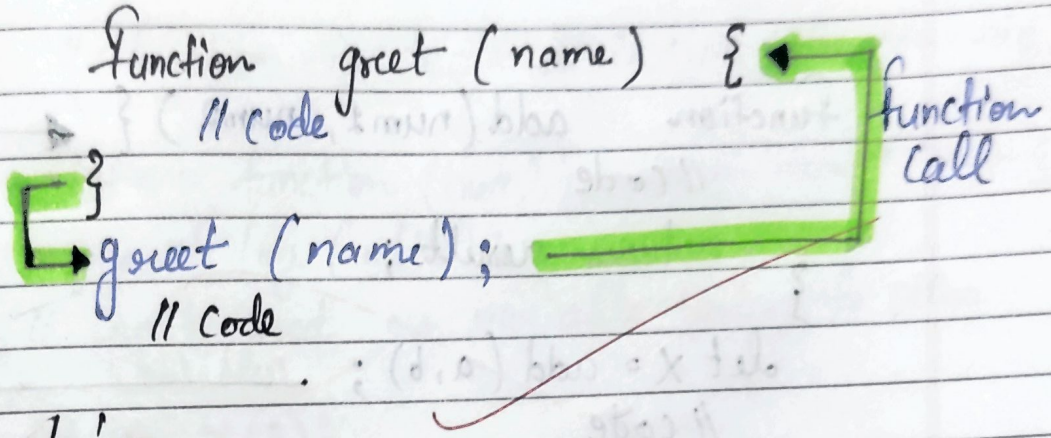
```
// Program to print a text
// declaring a function
function greet () {
    document.write ("Hello World");
}
greet ();
```

Output :-

Hello World

FUNCTION WITH PARAMETERS

- A function also be decalared with parameters.
- A parameter is a value that is passed when declaring a function.



Example :-

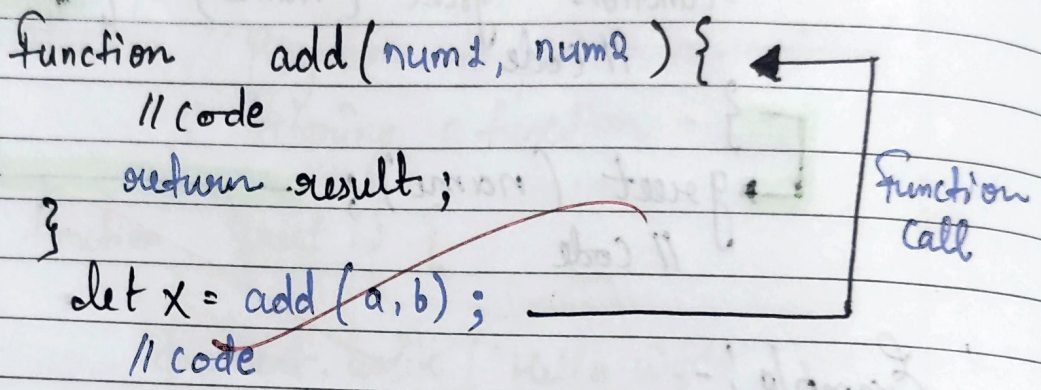
```
function greet (name)
{
document.write("Hello" + name + ":)");
}
// variable name can be different
let name = prompt ("Enter a name:");
// Calling function
greet (name);
```

Output :-

Enter a nam : Anshika Patel
Hello Anshika Patel :)

FUNCTION RETURN

The return can be used to return the value to a function call. It is denotes that the function has ended. Any code after return is not executed.



Example:- SUM OF TWO NUMBERS

```
function add(a,b){
    return a+b;
}
// take input from the user
let num1 = prompt("Enter 1st Number:"); parseFloat);
let num2 = prompt("Enter 2nd Number:"); parseFloat);
// calling function
let result = add(num1, num2);
// display the result
document.write("The Sum is" + result);
```

Output:- Enter ~~first~~ Number: 3.4
Enter 2nd Number: 4
The Sum is 7.4

FUNCTION EXPRESSION

Function can also be defined as expression.
for Example :-

```
// Program find the Square of a number

// function is declared inside the variable

let x = function (num) { return num * num };
console.log (x(4));
// Can be used as variable value for other
variables
let y = x(3);
console.log (y);
```

Output :- 16
9

Benefits of function Using

2) function make codes reusable.
2) function make Program easier as each small task is divided into a function.
3) function increase readability.

17/03/23