JavaScript

what is JavaScript?

- JavaScript is a programming language that adds interactivity to your website (for example games, responses when buttons are pressed or data is entered in forms, dynamic styling, and
- animation). JavaScript (JS) is a programming language mostly used to dynamically script webpages on the client side, but it is also often utilized on the server-side, using packages such as Node.js.
- JavaScript should not be confused with the Java programming language.

- Using JavaScript, we can:

1. Change HTML Content
2. Change HTML Attributes values
3. Change HTML Styles (CSS)
4. Change HTML Elements

- In HTML, JavaScript code must be inserted between <script> and </script> tags.

JavaScript files have the file extension **.js**. To use an external script, put the name of the script file in the src (source) attribute of a <script> tag.

You can place an external script reference in <head> or <body> as you like.

The script will behave as if it was located exactly where the <script> tag is located. External scripts cannot contain <script> tags.

Placing scripts in external files has some advantages:

- It separates HTML and code
- It makes HTML and JavaScript easier to read and maintain
- Cached JavaScript files can speed up page loads

Display outputs of JavaScript code:

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

Using document.write()

For testing purposes, it is convenient to use ***document.write()***

Using document.write() after an HTML document is loaded, will delete all existing HTML. The document.write() method should only be used for testing.

Using window.alert()

You can use an alert box to display data

Using console.log()

For debugging purposes, you can use the ***console.log()*** method to display data.

Using innerHTML

To access an HTML element, JavaScript can use the ***document.getElementById(id)*** method.

The id attribute defines the HTML element. The innerHTML property defines the HTML content.

JavaScript | Syntax

- **JavaScript** is the lightweight and dynamic computer programming language. It is used to create client side dynamic pages. It is open source and cross-platform language.
- **Basic Syntax:**
- <script> Document.write("hello"); </script>

- **JavaScript Variables:** A JavaScript variable is the simple name of storage location where data to be stored. There are two types of variables in JavaScript which are listed below:
- **Local variables:** Declare a variable inside of block or function.
- **Global variables:** Declare a variable outside function or with window object.

- **JavaScript Operator:** JavaScript operators are symbols that are used to compute the value or in other word we can perform operations on operands. Arithmetic operators (+, -, *, /) are used to compute the value and Assignment operator (=, +=, %=) are used to assign the values to variables.

JavaScript Expression: Expression is the combination of values, operators and variables. It is used to compute the values.

JavaScript Comments: The comments are ignored by JavaScript compiler. It increase the readability of code. It adds suggestions, Information and warning of code. Anything written after double slashes // (single line comment) or between /* and */ (multi line comment) is treated as comment and ignored by JavaScript compiler.

JavaScript Data Types: JavaScript provides different datatype to hold different values on variable. JavaScript is a dynamic programming language, it means do not need to specify the type of variable. There are two types of data types in JavaScript.

- Primitive data type
- Non-primitive (reference) data type

- **JavaScript Functions:** JavaScript functions are the blocks of code used to perform some particular operations. JavaScript function is executed when something call it. It calls many times so the function is reusable.

JavaScript Variables:

JavaScript Variables: JavaScript variables are containers for storing data values. In this example, x, y, and z, are variables

- All JavaScript variables must be identified with unique names. These unique names are called identifiers.
- Identifiers can be short names (like x and y) or more descriptive names (age, sum).
- The general rules for constructing names for variables (unique identifiers) are:
  - Names can contain letters, digits, underscores, and dollar signs.
  - Names must begin with a letter
  - Names are case sensitive (y and Y are different variables)
  - Reserved words (like JavaScript keywords) cannot be used as names
  - JavaScript identifiers are case-sensitive.

EXAMPLE:

```
var x = 5;
var y = 6;
var sum= x + y;
```

```
<html>
 <head>
  <title> Varibales in JavaScript</title>
 <meta charset="utf-8">
 <script type="text/javascript">
 var a= 100;  var b= 10.124;  var c="JAVASCRIPT";
var d = 'java Script'  var e = true;  var f = false;
console.log(a);
 console.log(b);
 console.log(c);
console.log(d);
 console.log(e);
console.log(f);
 </script>
 <</head>
 <body>
  <h1> Hello JavaScript </h1>
  <h2> This is my second Program in JavaScript</h2>
 </body>
</html>
```

•**Local Variable:** When you use JavaScript, local variables are variables that are defined within functions. They have local scope, which means that they can only be used within the functions that define them.
•**Global Variable:** In contrast, global variables are variables that are defined outside of functions. These variables have global scope, so they can be used by any function without passing them to the function as parameters.

•JavaScript Data Types:

• JavaScript variables can hold numbers like 100 and text values like "Harry Potter". In programming, text values are called text strings. JavaScript can handle many types of data, but for now, just think of numbers and strings. Strings are written inside double or single quotes. Numbers are written without quotes. If you put a number in quotes, it will be treated as a text string.

•**Datatypes:**

- Strings
- Numbers
- Boolean
- Array: var fruits = ["Apple", "Mango", "Banana”];
- Object: var employee = {firstName:"John", lastName:"Smith", age:50};
- Typeof
- Null
- Empty value

JavaScript Operators:

- Types of Operators in JavaScript :

1. Arithmetical Operators
2. Assignment Operators
3. Increment / Decrement Operators
4. Relational Operators
5. Logical Operators
6. Concatenation Operators

- 1)Arithmetical Operators : -

  \+ Addition
  \- Subtraction
  * Multiplication
  / Division
  % remainder

- 2)Assignment Operators : Used to assign value into variable

  = assigns to
  += add and assigns to
  -= Subtract and assigns to
  *= Multiply and assigns to
  /= Divide and assigns to
  %= Remainder assigns to

3)Increment / Decrement Operators:-

++ +=1 (or) Variable = Variable +1

-- -=1 (or) Variable = Variable -1

4)Relational Operators:-

== equals(comparition)

!= not equals

< less than

> Greater than

<= less than or equal to

>= greater than or equal to

5) Logical Operators :- Logical operators mainly used for combining two conditions.

&& □ and (both condition must be true)

|| □ OR (at least any one condition should be true)

! □ not (given condition will be reverse)

6) Concatenation Operator :- It is used to attach two strings (that means at the end of first string, the second string will be added).

number + number = addition

string + number = concatenation

number + string = concatenation

string + string = concatenation

JavaScript Functions:

JavaScript Functions:

- Function is a collection of statements, to perform some specific operation (work) .Function is a keyword, Function name is given any name, but (space not allowed), functionname doesn't allow the space, cannot be same as the keywords, and cannot contain the special characters
- but (underscore (_), dollar ($) symbols are allowed.
- Functionname must be started with alphabets only. Function can receive input values. Input values nothing but arguments.
- To receive the input values where part a create a variables. Variables nothing but parameters Parameters means variables to receive arguments,

ex: parameter1,parameter2….. Parameters are a optional, Result nothing but what return value

Syntax :

```
function functionName(parameter1, parameter2,…….) (parameters also a optional)
    {
          statements return (value); (return value is a optional
     }
    functionName(); //function call
```

```
)//type-1 function with return value

function test1(a)
{
  return a;
}
console.log("a1:",test1(10));

//type-2 function expression

var test2=function(a)
{
  return a;
}
console.log("a2:",test2(20));

//type-3 Arrow function

var test3=a=>a;
console.log("a3:",test3(30));
```

“ARGUMENTS” - PREDEFINED VARIABLE :- It is used to represent all the argument values that are received by the current function.
It is a pre-defined variable that can be used only in the functions.

- arguments : Returns collection of all argument values
- arguments[index] : Returns the single argument value based on given index
- arguments.length : count of argument values received

```
function sample(a,b)
      {
      console.log(arguments);
      console.log(a); // output : 10
      console.log(arguments[0]); // output : 10
      console.log(b); // output : 20
      console.log(arguments[1]); // output : 20
       console.log(arguments[6]); // output : Undefined means not decleared
      }
      Sample(10,20,30,40,50,60);
```

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>javascript</title>
    <link rel="stylesheet" href="javascript.css">
    <script src="javascript.js"></script>
</head>
<body>
       <h2 id="h">welcome to javascript</h2>
       <input type="button" value="changetoblue" onclick="changeBlue()">
       <input type="button" value="changetored" onclick="changeRed()">
       <input type="button" value="changecolor" onclick="changeColor()">
       <br><br>
        <img src="pics/img.jpg" id="i" alt="no img" height="70px" width="70px">
        <br>
        <input type="button" value="happy" onclick="changeHappy()">
        <input type="button" value="sad" onclick="changeSad()">
```

```
<script>
        function changeBlue()
        {
            document.getElementById("h").style.color="blue";
        }
        function changeRed()
        {
            document.getElementById("h").style.color="red";
        }
        function changeColor()
        {
            var c=prompt("Enter color","");
            document.getElementById("h").style.color=c;
        }
        function changeHappy()
        {
            document.getElementById("i").src="pics/html5.png";
        }
        function changeSad()
        {
            document.getElementById("i").src="pics/css3.png";
        }
 </script>
</body>
</html>
```

CONTROL STATEMENTS :

- **CONTROL STATEMENTS:**
- By default, all the statements executes in sequential order (top – to – bottom) In order to change (control) the execution flow of the statements. We have to use control statements.
- Types of Control Statements:-
- a. Conditional Control Statements (Conditional Operator (? :))
- b. Looping Control Statements
- **a. Conditional Control Statements** :- Conditional control statements divided into 2 types (things)
- 1. If
- 2. Switch case
- **b. Looping Control Statements:** Looping control Statements are 3 types (things)
- 1. While
- 2. Do-while
- 3. for

a. Conditional Control Statements:

I) "If" Conditional control statement:- If you want to check the condition first, and execute the statements only when the condition is TRUE, then use "if".

0. If (simple if)
1. If-else
2. Else-if
3. Nested if

if (simple if):- simple check the condition, if the condition is true only you want the execute the statements. The condition is false , Don't execute the statements.

Syntax:

```
if (expression)
{
Statement(s) to be executed if expression is true ;
}
```

If-else :-

Syntax :

```
if (expression)
 { Statement(s) to be executed if expression is true }
  else
{ Statement(s) to be executed if expression is false }
```

else-if :- The if...else if... statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions

Syntax:

```
if (expression 1)
{ Statement(s) to be executed if expression 1 is true }
else if (expression 2)
 { Statement(s) to be executed if expression 2 is true }
```

Nested if example:

```
if (a>=b)
{
      if (a==b)
       {
        console.log("a is equal to b");
       }
      else
      {
      console.log("a is greater than b");
      }
}
 else
    {
     console.log("a is less than b");
    }
```

2)Switch..Case : The objective of a switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

Syntax:

```
 switch (expression)
 {
case value1: statement(s) to be executed; break;
case value2: statement(s) to be executed; break;
...
default: statement(s);
break; //it is optional to write break, if default is the last statement in the switch block
 }
```

b.Looping Control Statement :-

The while Loop : The most basic loop in JavaScript is the while loop. The purpose of a while loop is to execute a statement or code block repeatedly as long as an expression is true. Once the expression becomes false, the loop terminates.

Note: the expression has to be true in the beginning to enter the loop. The syntax of while loop in JavaScript is as follows −

Syntax:

```
while (expression)
{
Statement(s) to be executed if expression is true
 }
```

The do...while Loop: The do...while loop is similar to the while loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is false.

The syntax: for do-while loop in JavaScript is as follows −

```
do
{
  Statement(s) to be executed;
}
while (expression);
```

for Loop : The 'for' loop is the most compact form of looping. It includes the following three important parts −

- ▫ The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- ▫ The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- ▫ The **iteration statement** where you can increase or decrease your counter.

Syntax:

```
for (initialization; test condition; iteration statement)
{
Statement(s) to be executed if test condition is true
}
```

JAVASCRIPT ARRAYS

- **The Array** object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.
- **Syntax:-** Use the following syntax to create an Array object –
- var fruits = new Array["apple", "orange", "mango"];
- Some important properties and methods of an Array:
- **1. length:** returns the length(size) of an Array
- **2. pop() :** Removes the last element from an array and returns that element.
- **3. push() :** Adds one or more elements to the end of an array and returns the new length of the array.
- **4. toString() :** converts an array to a string of (comma separated) array values.
- **5. join() :** joins all array elements into a string. it behaves just like toString(), but in addition you can specify the separator. E.g. fruits.join(" * ");
- **6. sort() :** sorts an array alphabetically in ascending order
- **7. reverse():** reverses the elements in an array. You can use it to sort an array in descending order

Objects in Javascript

- **Objects,** in JavaScript, is it's most important data-type and forms the building blocks for modern JavaScript. These objects are quite different from JavaScript's primitive data-types(Number, String, Boolean, null, undefined and symbol) in the sense that while these primitive data-types all store a single value each (depending on their types).

- **Creating Objects**:

There are several ways or syntax's to create objects. One of which, known as the Object literal syntax, we have already used. Besides the object literal syntax, objects in JavaScript may also be created using the constructors, Object Constructor or the prototype pattern.

1.Using the Object literal syntax : Object literal syntax uses the {…} notation to initialize an object an its methods/properties directly.

Let us look at an example of creating objects using this method :

```
var obj =
   {
       Key1 : value1,
       key2 : value2,
   };
```

These members can be anything – strings, numbers, functions, arrays or even other objects. An object like this is referred to as an object literal. This is different from other methods of object creation which involve using constructors and classes or prototypes, which have been discussed below.

2.Object Constructor : Another way to create objects in JavaScript involves using the "Object" constructor. The Object constructor creates an object wrapper for the given value. This, used in conjunction with the "new" keyword allows us to initialize new objects.

3.Constructors: Constructors in JavaScript, like in most other OOP languages, provides a template for creation of objects. In other words, it defines a set of properties and methods that would be common to all objects initialized using the constructor.

Example:

```
1.let school = {
      name : "Vivekanada",
      location : "Delhi",
      established : 1971,
      20 : 1000,
      displayinfo : function() {
            console.log(`${school.name} was established
            in ${school.established} at ${school.location}`);
      }
}
console.log(school.name);
console.log(school.established);
```

```
2.var person = {firstName:"John", lastName:"Doe", age:50,
eyeColor:"blue"};
```

JavaScript | Events

Javascript has events to provide a dynamic interface to a webpage. These events are hooked to elements in the Document Object Model(DOM).
These events by default use bubbling propagation i.e, upwards in the DOM from children to parent. We can bind events either as inline or in an external script.

There are some javascript events:

1) onclick events: This is a mouse event and provokes any logic defined if the user clicks on the element it is bound to.

2) onkeyup event: This event is a keyboard event and executes instructions whenever a key is released after pressing.

3) onchange event: This event detects the change in value of any element listing to this event.

4) onmouseover event: This event corresponds to hovering the mouse pointer over the element and its children, to which it is bound to.

5) onblur event: This event is evoked when an element loses focus.

JavaScript DOM

- Document Object Model
- The **document object** represents the whole html document.
- When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

JavaScript Form Validation

- **Form validation:-**normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button. If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process which used to put a lot of burden on the server. JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.
- ▫ **Basic Validation** – First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.
- ▫ **Data Format Validation** – Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.