**JAVASCRIPT**

**🡪Javascript** is used for the behavior of webpages.

🡪 it is the programming language of the web.

🡪 javascript can change the HTML Content using **innerHTML**

**Example**: <p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>

🡪 Javascript also change HTML Attributes.

**Example**:

<buttononclick="document.getElementById('myImage').src='pic\_bulbon.gif'">Turnonthe light</button>

<img id="myImage" src="pic\_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic\_bulboff.gif'"> Turn off the light</button>

🡪 JavaScript can also change the CSS Content

**Example**

<button onclick=’document.getElementById(‘demo’).style.font-size=25px’>Change Font</button**>**

**🡪** Javascript is also used to hide the HTML Element

**Example**

<button onclick=’document.getElementById(‘demo’).style.display=’none’>Click Me</button>

🡪 Javascript is also used to display the HTML Element

**Example**

<button onclick=’document.getElementById(‘demo’).style.display=’block’>Click Me</button>

**Where to code JavaScript**

🡪 Javascript will be written inside the <script>….</script>

**Example**

<script>

document.getElementById(‘demo’).innerHTML=”Java Script!”

</script>

🡪 JavaScript can be inserted in <head> or <body> or both.

**Function**

🡪 Function is a piece of code which is re-usable.

**Example**

<script>

function myFunc()

{

document.getElementById(‘demo’).innerHTML=”JavaScript”

}

</script>

<body>

<button onclick=”myfunc()”>Try It</button>

</body>

🡪 we can define the function constructor using built-in Javasscript method called function().

**Example**

var func=new function(“a”, “b”, return a\*b)

var x=func(3,4)

**Self-Invoking Functions**

🡪 Self calling of a function without being explicitly called.

🡪Function expression calls explicitly when it is enclosed by ()

🡪 You can’t call self invoking function.

**Example**

(function (){var x=”Hello”; })();

**Call()**

🡪 call() is the built-in method in Javascript.

🡪with call() object can used as method to call another method

**Example**

var person = {  
  fullName: function(city, country) {  
    return this.firstName + " " + this.lastName + "," + city + "," + country;  
  }  
}  
var person1 = {  
  firstName:"John",  
  lastName: "Doe"  
}  
person.fullName.call(person1, "Oslo", "Norway");

**Apply**

**🡪 apply()** is similar to the the call()

🡪 The main difference between the two is call() will take arguments separately whereas

apply() method will takes all arguments in the form of an array**.**

**Example**

var person = {  
  fullName: function() {  
    return this.firstName + " " + this.lastName;  
  }  
}  
var person1 = {  
  firstName: "Mary",  
  lastName: "Doe"  
}  
person.fullName.apply(person1);

🡪 we can also simulate apply to find max of array.

**Example:** Math.max.apply(Null,[1,2,3])

**External Script**

**🡪** we can also write JavaScript externally and include in our HTML Document.

🡪 Document Should be saved in ‘.js’ format.

**Example:** <script src="myScript.js"></script>

**Javascript Output**

🡪Writing into an HTML element, using innerHTML. it defines the HTML Element.

**Example:**<button onclick=’document.getElementById(‘demo’).innerHTML=’Bye!’>Click</button>

**document.write()**

🡪Writing into the HTML output using document.write().

**Example:**

<script>

document.write(5+6)

</script>

**Note:**

**1.** if this action performed, then the previous content of HTML Element will be removed.

2. document.write() should only used for testing.

**window.alert()**

🡪 Writing into an alert box, using window.alert().

**Example**

<script>

Window.alert(5+6)

</script>

🡪 we can also avoid “window” to get same result.

**Example**

<script>

alert(5+6)

</script>

**Console.log()**

🡪 Writing into the browser console, using console.log().

🡪 it is used in debugging the data in the web browser.

🡪 in web browser click F12 and click on debug,and again click on run to get the result.

**Example**

<script>  
console.log(5 + 6);  
</script>

**window.print()**

**🡪**window.print() method is used to print the content of current window.

**Example:** <button onclick="window.print()">Print this page</button>

**Javascript Program**

**🡪** A program is the list of instructions. Each instruction individually called as statement.

**Example**

var a=2;

var b=3;

var c=a+b;

🡪Add semicon(‘;’) at end indicates end of the instruction.

🡪 we can also write all instructions in a single statemet.where each statement is separated by ‘;’.Eg: a = 5; b = 6; c = a + b;

🡪 Javascript ignores the white spaces and the blank lines.

**Javascript Keywords**

**🡪** Keywords are the reversed words which can’t be defined in other terms.

🡪 few keywords are break, continue, debugger, do ... while, for, function, if ... else, return, switch, var.

**🡪** javascript supports arithmetic, logical, relational operators.

🡪fixed values are called as literals. **Eg:** a=2

🡪 variable values are called as variables. **Eg**: var a=2;

|  |
| --- |
|  |
|  |
|  |

**Comments**

**🡪** Comments are non-executable statements of the program.

🡪 Single line comments are represented in the form of // comment..

🡪 Multi line comments are represented in the form of /\*comment\*/

**Variables**

**🡪** variables are the named memory locations

🡪 Names can contain letters, digits, underscores, and dollar signs.

🡪 Names must begin with a letter

🡪 Names can also begin with $ and \_ (but we will not use it in this tutorial)

🡪 Names are case sensitive (y and Y are different variables)

🡪 Reserved words (like JavaScript keywords) cannot be used as names

**🡪** Javascript supports number and string data types.

🡪 Re-declaration of variables is possible in the Javascript.

**Let**

🡪”**Let**” is used to point the variables at local scope.

**Example**

Var x=8

{

let x=2; //in this scope value of x is 2.

}

**Note**

🡪Using a let variable before it is declared will result in a ReferenceError.

🡪 Redeclaring a let variable with var, in the same scope, or in the same block, is not allowed

**Example**

let x = 2;       // Allowed  
var x = 3;       // Not allowed  
  
{  
  let x = 4;   // Allowed  
  var x = 5;   // Not allowed  
}

**Cons**t

🡪 The usage of **const** is as same as **let** with a difference that const can’t be re-assigned**.**

**Note**

**🡪 const** must be initialized at the time of declaration**.**

**Example:**

const pi=3.142434 (Correct)

const pi

pi=3.142434 (in-correct)

**🡪**Constant objects can be changeable Whereas re-assigning is not possible.

**Example**

const car = {type:"Fiat", model:"500", color:"white"};  
car.color = "red";  
car.owner = "Johnson”**(correct)**

const car = {type:"Fiat", model:"500", color:"white"};  
car = {type:"Volvo", model:"EX60", color:"red"};    // ERROR

🡪 Constant Arrays can change like changing the values at index level and add additional elements.

**Example**

Const arr=[‘volvo’,’BMW’,’Benz’];

arr[0]=’bugati’

arr.push(“Lexus”) **(correct)**

const cars = ["Saab", "Volvo", "BMW"];

cars = ["Toyota", "Volvo", "Audi"];    // ERROR

🡪 **Typeof** is used to know the type of the Javascript variable.

**Undefined**

**🡪** variable without declaration is called as undefined.

**Example**

**var car** //here the variable car is declared but undefined.

**Note:** The declaration var str=”” is a string, not undefined**.**

**Javascript Funtction**

**🡪** Function is a block of code which is re-usable.

**Syntax**

Function function\_name(variable 1, variable 2)

{

Block of code;

return <variable-name>(optional);

}

**Javascript Objects**

**🡪**Objects are real world entities. Objects have the properties.

**Example**

Var car={brand:”fiat”,model:500,mode:”petrol”}

**(or)**

var person = {

firstName: "John",

lastName: "Doe",

age: 50,

eyeColor: "blue"

};

**🡪** Accessing the properties can be achieved by object.property (or) object[property].

**Note:** Avoid Declaring integers, strings, Boolean as objects as they increase complication and slows down the execution speed**.**

**Javascript Strings**

**🡪** Array of characters together can be called as string.

**Syntax:** var x=”Hello” or var x=’hello’ (both single and double quotes are acceptable).

🡪we can use double quotes within single quotes or vice-versa

**Example:var sr=”hello ‘I am vishnu’” or var sr=’hello “I am vishnu”’**

**Escape characters**

**🡪**for example, var str=”hello “I am Vishnu” from vidyanikethan”. in this example, string will be chopped to hello as compiler misunderstanding**.**

**🡪 To avoid this, we will use backslash quote.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Escape Sequence** | **Result** | **Description** | **Example** |
| \’ | ‘ | Single Quotation | var x = "We are the so-called \’Vikings\’ from the north."; |
| \” | “ | Double Quotation | var x = "We are the so-called \”Vikings\“ from the north."; |
| \\ | \ | Backslash | var x = "We are the so-called Vikings\\ from the north."; |
| \b | Backspace |  | Var x=”hello \b world” |
| \f | formfeed |  |  |
| \n | Newline |  | var=”new\ngame” |
| \r | Carriage return |  |  |
| \t | Horizontal tab |  | Var=”Hello\tWorld”. |
| \v | Vertical tab |  | Var x=”Hello\vWorld” |

**Strings as Objects**

**🡪** we can also declare strings as objects.Eg:var x=new string(“Hello”)

**Comparing Strings and objects**

**🡪** using **==** will compare the values

🡪using === will leads to error because, in === both type and value will gets checked

**Example**

var x=”Hello”

var y=new string(”Hello”)

x==y //returns True because having same values.

String operations

🡪**length** is used to find the lemgth of the string.

**Example**

var x=”hello”

var y=x.length;

🡪 **indexof** is used to get the index of first occurance of the string.

**Example**

var x=”Hello I am Vishnu”

var y=x.indexof(“Vishnu”)

🡪 **lastindexof** is used to get the index of last occurance of the string.

**Example**

var x=”Hello I am Hello”

var y=x.lastindexof(“Hello”)

🡪**search** will find the substring in the main string.

**Example**

var x=” Hello I am Vishnu”

var y= x.search(“Vishnu”)

**Note:** The search() method cannot take a second start position argument whereas  indexOf() method cannot take powerful search values (regular expressions).

**Cutting a Substring**

**🡪** slicing means cutting a substring from main string**.**

**🡪** There are three methods in extracting a substring.

**slice(*start*, *end-1***)🡪slice(4,8)—extracts from 4 to 7 index character.negative index also starts from end.if we specify one argument then it extracts all characters from begin index.

**substring(*start*, *end*)🡪**same as slice and didn’t accept negative values.

**substr(*start*, *length***)🡪 substr() is similar to slice().

**Example**

var str = "Apple, Banana, Kiwi";  
var res = str.substr(7, 6);

**Output:** Banana

The difference is that the second parameter specifies the **length** of the extracted part. if you omit the second parameter, substr() will slice out the rest of the string.

**Example**

var str = "Apple, Banana, Kiwi";  
var res = str.substr(7);

The result of res will be:

Banana, Kiwi

**String replace**

**🡪replace** is used to replace the content of the string with another part

**Example**

Var str=”Hello, I am Vishnu”

Str.replace(“Vishnu”,”avinash”)

**🡪** To replace case insensitive, use a **regular expression** with an /i flag (insensitive):

**Example**

str = "Please visit Microsoft!";  
var n = str.replace(/MICROSOFT/i, "W3Schools");

**🡪** To replace all matches, use a **regular expression** with a /g flag (global match):

**Example**

str = "Please visit Microsoft and Microsoft!";  
var n = str.replace(/Microsoft/g, "W3Schools");

**Converting uppercase to lowercase and lowercase to uppercase**

**🡪 toUpperCase** is usesd to convert from lowercase to uppercase**.**

**Example**

var str=”hello”

str.toUpperCase() //HELLO

🡪 **toLowerCase** is usesd to convert from uppercase to lowercase**.**

**Example**

var str=”HELLO”

str.toUpperCase() //hello

**Concatination**

**🡪** Concatinationmeans joining two or more strings together**.**

**🡪 ‘+’** or **‘concat’** is used to join the two strings.

**Example**

var x=”hello”+” world”

var x=”hello”.concat(“ “,”world”)

**trim()**

🡪 it is used to remove the white spaces before and after the string.

**Example**

Var x=” Hello”

alert(str.trim()) //prints Hello

**String Padding**

**🡪** padding is used to add zeros before or after the string**.**

**🡪** padStart() and padEnd() are the two methods**.**

**Example**

let str = "5";  
str = str.padStart(4,0);  
// result is 0005

let str = "5";  
str = str.padEnd(4,0);  
// result is 5000

C**harAt()**

**🡪 charAt() is** used to find the character present in the string based din the index.

**Syntax:** string.charAt(index)

**Example**

var x=”hello”

x.charAt(0) //prints ‘h’

C**harCodeAt()**

**🡪 charCodeAt() is** used to return the Unicode of the character present in the string based din the index.Unicode ranges from 0 to 65535.x

**Syntax:** string.charCodeAt(index)

**Example**

var x=”hello”

x.charCodeAt(0) //prints ‘h’

**Convert String to Array**

🡪we can convert array of characters using “.split(‘seperator’)

Example

var x=”a,e,i,o,u”

var y=x.split(‘,’)

alert(y) //[‘a’,’e’,’I’,’o’,’u’]

Number:

🡪 Addition of string with number or number with string or string with string will results string concatenation.

**Example**

var x = 10;  
var y = 20;  
var z = "30";  
var result = x + y + z; //output wil be 3030

**Explanation:** Javascript works from left to right as x and y are integers results in addition of two numbers which gives 30 and as z is is string would results in string concatenation and results 3030

🡪subtraction, multiplication, division of two integer strings will perform actual arithematic operations

**Sorting Numbers in array**

🡪By-default strings can be sorted. For making numbers sorted use the methodology below

**Example**

var points = [40, 100, 1, 5, 25, 10];  
points.sort(function(a, b){return a - b});

**Explanation**

**🡪** When the sort() function compares two values, it sends the values to the compare function, and sorts the values according to the returned (negative, zero, positive) value.

🡪 if the result is negative a is sorted before b.

🡪 If the result is positive b is sorted before a.

🡪 If the result is 0 no changes are done with the sort order of the two values.

**Example**

<button onclick="myFunction1()">Sort Alphabetically</button>  
<button onclick="myFunction2()">Sort Numerically</button>  
  
<p id="demo"></p>  
  
<script>  
var points = [40, 100, 1, 5, 25, 10];  
document.getElementById("demo").innerHTML = points;  
  
function myFunction1() {  
  points.sort();  
  document.getElementById("demo").innerHTML = points;  
}  
function myFunction2() {  
  points.sort(function(a, b){return a - b});  
  document.getElementById("demo").innerHTML = points;  
}  
</script>

**Sorting Array object**

**🡪** Array objecys can be soted based on the propery and property can be accessed using **.** operator.

**Example:**

var cars = [  
  {type:"Volvo", year:2016},  
  {type:"Saab", year:2001},  
  {type:"BMW", year:2010}  
];

cars.sort(function(a, b){return a.year - b.year});

**for-each iterator**

🡪 **forEach** is used to iterate over the containers like arrays or associate arrays.

**Example**

var txt = "";  
var numbers = [45, 4, 9, 16, 25];  
numbers.forEach(myFunction);  
  
function myFunction(value, index, array) {  
  txt = txt + value + "<br>";  
}

Array.map()

🡪 map will creates a new array and perform the operation.

**Syntax:** var new\_array=array.map(operation)

**Example**

var arr=[1,3,34,33]

var new\_arr=arr.map(myfunction)

Function myfunction(value, index, array)

{

return value+2

}

Array.filter()

🡪 filter eliminate the values based on conditions.

**Syntax:** var new\_array=array.filter(operation)

**Example**

var arr=[1,3,34,33]

var new\_arr=arr.filter(myfunction)

Function myfunction(value, index, array)

{

return value>10

}

Array.reduce()

🡪 reduce will perform the operation and make it single value.

**Syntax:** var new\_array=array.map(operation)

**Example**

var arr=[1,3,34,33]

var new\_arr=arr.map(myfunction)

Function myfunction(total, value, index, array)

{

return total+value

}

Array.every()

🡪 every will check each and every element in array and returns true, if all values are satisfies the condition, else returns false.

**Syntax:** var new\_array=array.every(operation)

**Example**

var arr=[1,3,34,33]

var new\_arr=arr.filter(myfunction) // returns False.

Function myfunction(value, index, array)

{

return value>10

}

Array.some()

🡪 some will check each and every element in array and returns true, if some values are satisfies the condition, if none of the element satisfies the condition. it returns false.

**Syntax:** var new\_array=array.some(operation)

**Example**

var arr=[1,3,34,33]

var new\_arr=arr.some(myfunction) // returns True.

Function myfunction(value, index, array)

{

return value>10

}

Array.findIndex()

🡪 findIndex will retuen the first index of array which passes the condition.

**Syntax:** var new\_array=array.findIndex(operation)

**Example**

var arr=[1,3,34,33]

var new\_arr=arr.findIndex(myfunction) // returns 2

Function myfunction(value, index, array)

{

return value>10

}

**Date Object**

🡪 **new date()** will be used to create date objects with current name and date.

🡪There are four ways to create date object

new Date()  
new Date(year, month, day, hours, minutes, seconds, milliseconds)  
new Date(milliseconds)  
new Date(date string)

🡪 6 numbers specify year, month, day, hour, minute, second

**Example**

var d = new Date(2018, 11, 24, 10, 33, 30);

🡪5 numbers specify year, month, day, hour, minute

**Example**

var d = new Date(2018, 11, 24, 10, 33);

🡪 4 numbers specify year, month, day, hour.

**Example**

var d = new Date(2018, 11, 24, 10);

🡪 3 numbers specify year, month, day

**Example**

var d = new Date(2018, 11, 24);

🡪 2 numbers specify year, month

**Example**

var d = new Date(2018, 11);

🡪you can’t omit month.specifying 1 attribute will results in milliseconds.

**Example**

var d = new Date(2018)

**Previous Century**

**🡪** previous century can be interpreted as 19XX.

**Example**

var d = new Date(99, 11, 24);

**new date(String)**

**🡪creates new object from string.**

**Example:**

var d = new Date("October 13, 2014 11:13:00");

Displaying dates

🡪**document.getElementById("demo").innerHTML = d.toString();** usedto display whole date and time

🡪**toUTCString()** used to convert date to UTC String**.**

**Syntax:**document.getElementById("demo").innerHTML = d.toString();//Fri, 28 May 2021 07:38:04 GMT

🡪**toDateString** used to convert date to datestring

**Syntax:**document.getElementById("demo").innerHTML = d.toString();//Fri, 28 May 2021

🡪 ToISOString is used to convert date to ISO Format.

**Javascript ses**

**🡪 A class is collection of methods and**

**🡪 class** keyword is used to create class and **constructor()** is always added.

🡪 **constructor** is used to initialize the members of the class.

**Example**

class Car {  
  constructor(name, year) {  
    this.name = name;  
    this.year = year;  
  }  
  age() {  
    let date = new Date();  
    return date.getFullYear() - this.year;  
  }  
}  
  
let myCar = new Car("Ford", 2014);  
document.getElementById("demo").innerHTML =  
"My car is " + myCar.age() + " years old.";

**Inheritance**

🡪inheritance is the process of acquiring characteristics from parents classes.

**Example**

Class car

{

Constructor(brand)

{

this.brand=brand;

}

Present()

{

return “I have this +this.brand;

}

}

class Model extends Car

{

Constructor(brand,model)

{

Super(brand);

This.model=model;

**}**

Getname()

{

return this.model;

}

Setname(new){

This.model=new;

}

static show() {  
    return this.present() + ', it is a ' + this.model;  
  }  
}  
  
let myCar = new Model("Ford", "Mustang");  
document.getElementById("demo").innerHTML = myCar.show();

JavaScript Methods

**Finding Elements**

**🡪**finding elements by id**-document.getElementById(‘id name’).innerHTML**

**🡪** finding elements by Tag name**-document.getElementByTagName(‘p’)**

**🡪** finding elements by class name**-document.getElementByClassName(‘p’).**

🡪finding elements by querySelectorAll()-**document.querySelectorAll("p.intro");**