PAGE EDGE
Date : 13 / 03 / 23

java Script

javaScript is user friendlyness

Q. History of javaScript :-
"Brenalen Eich" he is the father of javaScript introduced in 1995.
The first Name of javaScript is `MoCHA' → LiveScript → javaScript

Q. what is javaScript :-
1. javaScript is highlevel and object oriented and functional base programming language.
2. java Script is interpreted programing language
↓ line by line execution.
3. java Script is userfriendly programing language.

Differences between java and javaScript.

java	javaScript
java used inside only middle layer	javaScript used inside 3 layers
java is Not userfriendly	javascript is a userfriendly
java have Compiler	javaScript doesn't have Compiler
java is a multithread	javaScript is Single thread.
S.O.pln();	Console.log(); and document.write()

printing statements :- of javaScript
document.write();
Console.log();
In javaScript we have Two printing statements :-
1. document.write();
2. Console.log();

14-03-23

Html:- hypertext markup language. It is written inside the <body> tag.

CSS:- Cascading style sheet. It is written inside the <head> after the <title> tag <style>.

Java script:- Java script Code written inside anywhere but we suggest to write inside <body> using <script>.

* Document.write():

document.write() is a printing statement which used to see the output in the web browser. (client side) (<script>)

* Console.log():

Console.log(); is a printing statement we can see the output in the Console window. (Developer side).

** Operator

operator	Data	Datatype
==	✓	×
===	✓	✓

== is Compare the data

=== is Compare the data along with Datatype

* Keywords:-

→ keywords are reserved words which has predefined meaning in java script there is no fixed size keywords. we have Top → 49 keywords.

E.S → Top → 49 keywords.

↳ Echma script

* Variables:-

Variables are Containers which holds data.

It is also known has data holder

* Syntax to Declare a Variable:-

In java script we can declare variable in 4 ways:-

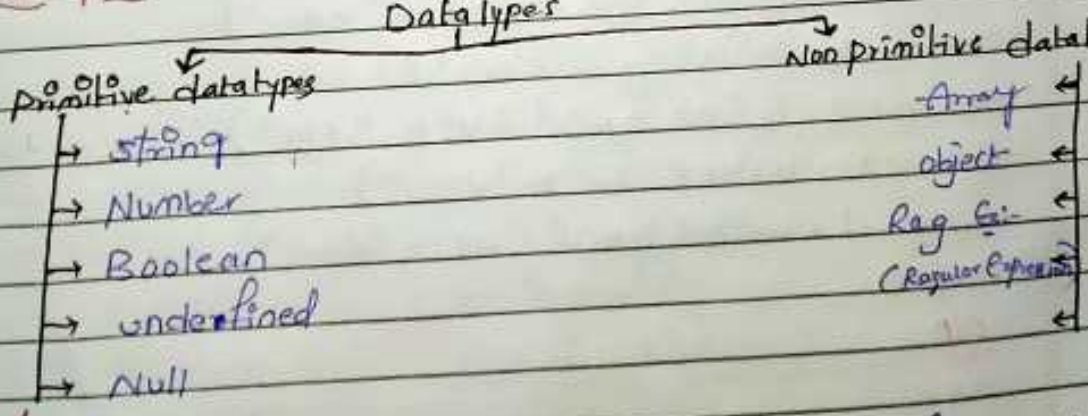
1. VaariableName = data; varName = data;
2. Varaw Variablename = data; (var) varName = data;
3. let VarName = data; (let) varName = data;
4. Const VariableName = data; (Const) varName = data;

Example

```
<body>
<script>
    x = 100;
    var name = "javaScript";
    let subject = "js";
    Const C1 = "x1";
→ document.write(x + "<br>");
→ Console.log(x);
→ document.write(name + "<br>");
→ Console.log(name);
→ document.write(subject + "<br>");
→ console.log(subject);
→ document.write(c1 + "<br>");
→ Console.log(c1);
</script>
</body>
```

p/o/p

Datatypes :-



String :-

A string is a set of character or a bunch of character.

In JavaScript We can represent a string in 3 ways

1) using double quotes (" ")
2) using single quotes (' ')
3) using back-Ticks (` `) → above the tab.

Ex:- →

```
Var firstname = "Sai";
Var lastname = 'kumar';
var fullname = 'Saikumar';

document.write(firstname + "<br>");
document.write(typeof(firstname) + "<br>");
document.write(lastname + "<br>");
document.write(type of(lastname) + "<br>");
document.write(fullname + "<br>")
document.write(typeof(fullname) + "<br>")
```

Number :-

whenever we want to store number kind of data, internally we make use of Number data type.

```
<script>
    Var x=100;
    Var y=78.996;
    Var Sal=99976257298790;
document.write(x + "<br>")
document.write(type of(x) + "<br>")
document.write(y + "<br>")
document.write(typeof(y) + "<br>")
document.write(Sal + "<br>")
document.write(typeof(sal) + "<br>")
</script>
```

O/P

```
100
number
78.996
number
99976257298790
bigint
```

Boolean :-

Whenever we have decision making Scenario we make use of boolean datatype.

It can return Two Values true or false.

Ex:-
```
<script>
    var x=true;
    document.write(x+"<br>");
    document.write(typeof(x));
</script>
```
o/p
true
boolean

Undefined :-

Declare a variable without assigning value to it undefined is the default value of any container.

Java
```
char xyz;
int x;
string x1;
double y;
```

Javascript
```
<script>
var x=84.39;
document.write(x+"<br>");
document.write(typeof(x));
</script>
```

Null :-

Null refers to nothing or Empty.

Whenever we want to make any ~~Condition~~ Container as a Empty Container we have to pass Null datatype as a data.

Ex:-
```
<script>
    var x=1234;
    var x=null;
    document.write(x+"<br>");
    document.write(typeof(x));
</script>
```
o/p
null
Object -> why Intrinsic ...

Special values :-
```
<script>
    document.write(100/0+"<br>");
    document.write(-100/0+"<br>");
    document.write("java"/23)
</script>
```
o/p
Infinity
-Infinity
Not a No (NaN)

16-8-23

we go for function reducing the lines of code & Reusability.

FUNCTION :-

General def :- Set of instructions given to the machine to perform Specific Task.

function intervals of js :- In order to execute same set of code again and again according to the user-requirement.

Syntax to Declare the function :-

```
function functionName()
{
    // logics or statements
}
```

(scope of the function)

Ex:- Advantage of function

* overCome the drawbacks like No of lines, redundency and

```
<body>
<script>
function qspiders()
{
    document.write("praveen" + "<br>")
    document.write("Ramya" + "<br>")
    document.write("Sruthi" + "<br>");
}
qspiders(); // function call statement.
</script> qspiders();
</body> qspiders();
</html> qspiders();
```

(keyword) (function name)

Ex:-

Drawbacks of functions is we get same output.

```
function qspidersStudent()
{
    document.write("student name is Rai" + "<br>")
    document.write("Rai is from A.P" + "<br>")
    document.write("<br>");
}
qspidersStudent()
qspidersStudent()
```

From the above example we note that for each and every function call we are getting same output which will lead to hardcoded value to overcome this drawback we are going for the concept of parameterised function.

parameterised function:-

Syntax:-

```
Function FunctionName( para1, para2, ..... paraN )
                        → parameter
{
    // statements or logic
}
FunctionName( args1, args2, ..... argsN ); // function call
```

parameter:-

parameter is a container which is capable of holding an argument.

argument:-

The data which passed in function call.

hard code output:-

hard code output means same output getting more than 1 time

dynamic output:-

dynamic output means different output getting more than 1 time

Ex:-

```
function wspidersStudent( name, place )
{
    document.write( "student name is " + name + "<br>" );
    document.write( "student is from " + place + "<br>" );
    document.write( "<br>" );
}
wspidersstudent( "Sai", "Andhra" );
wspidersstudent( "Manjula", "Delhi" );
wspidersstudent( "Suraj", "up" );
```

17/03/23

```
<script>
Ex:- function display(A, B, C)
{
  document.write("A value is " + A + '<br>')
  document.write("B value is " + B + "<br>")
  document.write("C value is " + C + "<br>")
  document.write("<br>");
}
display(300, 200, 100); // function call
display("sai"); // function call
display(true, 300); // function call
display(); // function call
</script>
```

O/P
300
200
100
sai
undefined
undefined
true
300
undefined
undefined
undefined
undefined

Note:-

→ from the above example we Note that in parameterised function data checking will Not happen.

→ passed parameter Need Not to be same as Passed argument.

* what if ~~parameter~~ I have arguments more than the parameter?

```
function display(A, B, C)
{
  document.write("A value is " + A + "<br>")
  document.write("B value is " + B + "<br>")
  document.write("C value is " + C + "<br>")
  document.write(arguments[3]);
  document.write("<br>");
}
display(100, 200, 300, 400, 500) // function call
```

A = 100
B = 200
C = 300

Arguments object

100	200	300	400	500
0	1	2	3	4

in the form of array

O/p
A value is 100
B value is 200
C value is 300
→ 400

PAGE EDGE
Date : / /

Inbuilt functions

User input :-

① prompt :- prompt is a function which is used to take User-input.

Syntax :- prompt (string message, [default value])

Ex :-
```
<script>
var age = prompt ("enter your age", 18)
document.write (age);
</script>
```
↳ default value presented in the console or

② alert() :-

alert() is a function which is used to throw the warning messages to the user.

Syntax :- alert ("message");

Ex :-
```
<script>
alert ("This page is Not for your use");
</script>
```

③ Confirm() :-

if want to take Confirmation message from the User we have to make use of Confirm ~~messeg~~ function.

Syntax :- Confirm ("message")

Ex :-
```
<script>
Confirm ("are you sure you want to visit this website");
</script>
```

o/p

[OK] [Cancel]

Window :-

window is supermost object in javaScript it consisting of 'n' no. of inbuilt functions, object, properties.

Functions :- f

Object :- { }

Properties :- :

Scope :-

Scope refers to boundary or visibility.

In javaScript we have 2 scope :-

1. Global Scope
2. Local Scope

* Global Scope Variable:-

A variable which is declare out-side the function Scope but inside the <Script> Tag we call it as Global Variable.

The Global variable can be Accessed Anywhere inside the <Script> Tag.

Ex:-

```
<script>
    x = 100;
  Var x1 = 200;
   let x2 = 300;
   Const x3 = 400;
  function xyz()
  {
    document.write("x value is " + x + "<br>"); // ---- 100
    document.write("x1 value is" + x1 + "<br>"); // ---- 200
    document.write("x2 value is" + x2 + "<br>"); // ---- 300
    document.write("x3 value is" + x3 + "<br>"); // ---- 400
    document.write("<br>");
  }
    xyz(); // function call
    document.write(x); // 100
    document.write(x1); // 200
    document.write(x2); // 300
    document.write(x3); // 400
</script>
```

inside the function

inside the script

* Local Scope Variable:-

A variable which is declared within the function Scope we call it as Local scope variable.

The Accessibility of Local Scope variable only with in the function Scope.

Ex:- →

VarName = data; Whenever there are Functionality VarName = data; is Considered as Global variable

PAGE EDGE
Date : / /

<Script>

Ex:- function xyz()
{
x = 100; // Global variable
var x1 = 200; // local
let x2 = 300; // local
const x3 = 400; // local
document.write("inside the function" + x + "
"); // 100
document.write("inside the function" + x1 + "
"); // 200
document.write("inside the function" + x2 + "
"); // 300
document.write("inside the function" + x3 + "
"); // 400
}
xyz(); // Function Call
document.write(x + "
"); // 100
document.write(x1 + "
");
document.write(x2 + "
");
document.write(x3 + "
");

Global variable
local variable

Var Name = data;
var Var Name = data;
let Var Name = data;
Const Var Name = data;

if you want to decla
re a local variable -
you must go with var,
let, Const variables.
inside the function.

</Script>

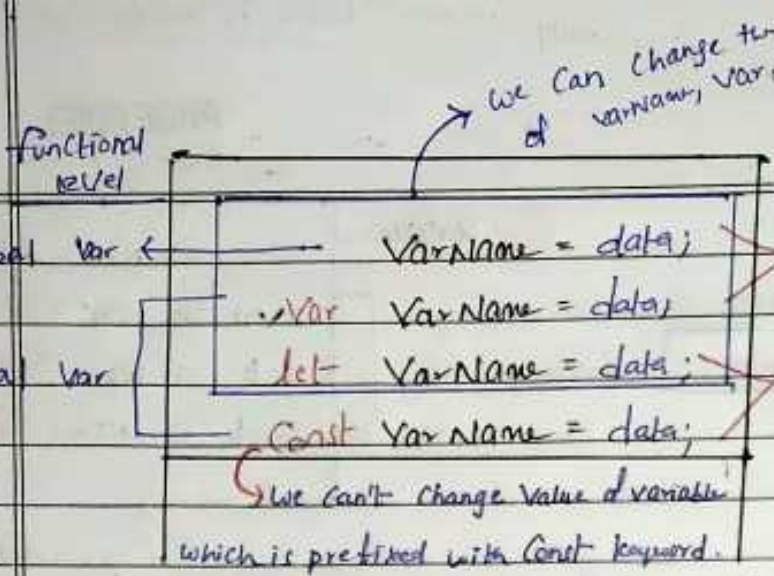
let, Const = data; Conditional statements

whenever there are Conditional based like if, Nested if, while, for and switch inside this if we want declare a local variable we go with let and Const.

Ex:- if (true)
{
x = 100; // Global
var x1 = 200; // Global
let x2 = 300; // local
Const x3 = 400; // local
document.write(x); // 100
document.write(x1); // 200
document.write(x2); // 300
document.write(x3); // 400
}
document.write(x); // 100
document.write(x1); // 200
document.write(x2); document.write(x3);

inside the Condition blocks

VarName = data; } Global
Var VarName = data; }
let VarName = data; } local
Const VarName = data; }



x = 100;
change x = 200; ✓

var x = 100;
var x = 200; ✓

~~var~~ let x = 200;
let x = 300; ✓

Const x = 200;
Const x = 300;
Error

Calculator — html and css

css part

```
<style>
    div{
        padding : 10px
        border : 2px solid red;
        width : 50%;
        margin: auto;
        text-align : center;
    }
</style>
</head>
```

html part:-

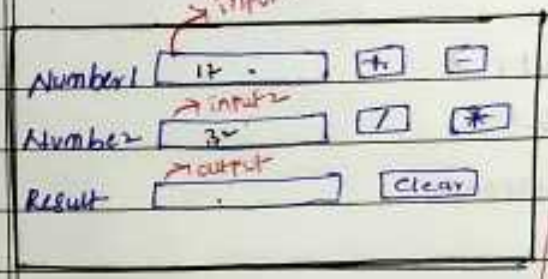
```
<body>
<div>
<h1> Calculator </h1>
<label for = " ">number1 </label>
<input type ="number" id = "input-1" >
<button> / </button>
<button> * </button>
<br> <br>
```

onclick = "call the function"

```
<label for = "">number2 </label>
<input type = "number" id = "input-2">
<button> + </button>
<button> - </button>
<br> <br>
<label for = "" > result </label>
<input type = "number"  id = "output">
<button> clear </button>
<br> <br>
</div>
</body>
```

Calculator

HTML



API (DOM) → BOM
(Application Programming interface)

Document object Model
↓ ↓ ↓
Html file | Tag element Node

① → document.getElementById("input1");
id name

x = 17
y = 3
z = x + y

DOM
document.getElementById();
↓ ↓
Html Tag

```
function ADD()
{
var x = document.getElementById("input1").value;
var y = document.getElementById("input2").value;
var z = parseInt(x) + parseInt(y)
document.getElementById("output").value = z;
}
```

Total Code →

```
<style>
 div{
     padding : 10px;
     border : 2px solid red;
     width : 50%;
     margin : auto;
     text-align : center;
   }
</style>
</head>
<body>
 <div> <h1> Calculator </h1>
  <label for=" "> number1 </label>
  <input type="number" id="input-1">
 <button onclick="ADD()"> + </button>
 <button onclick="SUB()"> - </button>
 <br> <br>
 <label for=" "> number2 </label>
 <input type="number" id="input-2">
 <button onclick="DVD()"> / </button>
 <button onclick="MLT()"> * </button>
 <br> <br>
 <label for=" "> result </label>
 <input type="number" id="output-">
 <button onclick="CLS()"> clear </button>
 <br><br>
</div>
 <script>
  function ADD()
  {
    var x=document.getElementById("input-1").value
    var y=document.getElementById("input-2").value
    var z = parseInt(x) + parseInt(y)
    document.getElementById("output").value=z;
  }
```

CSS

HTML

JS

SUB, DVD, MLT

```
Function CLS()
{
  document.getElementById("input1").value = Null;
  document.getElementById("input2").value = null;
  document.getElementById("output").value = null;
}
</Script>
</body>
</html>
```

OTP Generation

Math.random() function :-

random() is a function which is used to generate random numbers b/w the range 0 - 1 (floating value).

Eg:- Var x = Math.random()
Console.log(x); // 0.9198076

To generate 4 digit Number :-

∴ Math.random() * (HN - LN) + LN

where HN refers to Highest Number LN refers to Lowest Number.

Eg:-
Var result = Math.random() * (9999 - 1000) + 1000
Console.log(result); // 4239.6f89123

To remove floating Value or decimal Value :-

Math.floor() :-

floor() function is used to remove floating data.

Eg:-
Var x = Math.random() * (9999 - 1000) + 1000
Var otp = Math.floor(x);
Console.log(otp); // 7869

↓

document.getElementById("output").innerHTML = otp;

The above mentioned code is used to target html element and get inside the targeted element.

Note:-
if we want to target the html element we can use the .value only for <input typ> except input type we can use .innerHTML like <h1> <label> etc....

```
function genOtp()
{
  Var x = Math.random()*(9999-1000)+1000
  var otp = Math.floor(x);
  Console.log(otp)
  Var name = document.getElementById("username").value
  document.getElementById("output").innerHtml =
                    "dear "+name+"your Otp is "+otp;
}
```

CSS :- Code for OTP Generate :-

```
<style>
    div{
        padding:10px;
        border:2px solid red;
        text-align: center;
    }
    .output{
        border:2px solid black;
        background color: green;
        border radius: 20%;
    }
</style>
```

Html code :-

```
<body>
  <div>
    <label for=" " > username </label>
    <input type="text" id="name" >
    <br><br>
    <label for=" " > PhoneNumber </label>
    <input type="number" id="phone" >
    <br><br>
    <button onclick="genOtp()" > Send Otp </button>
    <br><br>
    <h1 id="output" ></h1>
</div>
```

Script Code :-

```
<Script>
    function genOtp()
    {
        Var x = Math.random() * (9999 - 1000) + 1000
        var otp = Math.floor(x);
        // Document.getElementById("output").innerHtml = otp;
var name = document.getElementById("username").value
// document.getElementById("output").innerHtml = "dear" + name + "your otp
                                                    is" + otp ;
    </Script>
    <body>
```

24/03/23

Return type function :-

Syntax :

function functionName()	function functionName()	function functionName
{	{	{
return data;	return var;	return Expression;
}	}	}
Returning Data;	Returning Variable;	Returning Expression;
Ex :- <Script>	Ex :- <script>	<script>
function display()	function display()	function display()
{	{	{
return 566;	Var x = 10 + "java";	return 10+89/7 * 69+68
}	return x;	}
Var x = display();	}	Console.log(display());
document.write(x);	Var y = display();	Var x = display();
</Script>	document.write(y);	document.write(x);
o/p	</Script>	</Script>
566	o/p	o/p
	10java	67.023 — Console
		67.023 — webpage

Advance Level very important Es-6

PAGE EDGE
Date : / /

⑧ Arrow function :-

Syntax :- Var Varname = ( ) ⇒ { logic }

Rules to write Arrow function :-

1. In Arrow() function NO Need of writing function Name.
2. In Arrow() function NO Need of writing function keyword.
3. In Arrow() function we can Neglet flower brasis { } only when we have one printing statement.
4. In Arrow() function we can Neglet paranthasis [()] in Arrow function only when we have One parameter → (name) and also we can Neglet paranthasis when we don't have any parameter but it must be replaced with (_) underscore. as shown in example 2.
5. we can write return type function even without return keyword as soon in example-5.

Examples

① Var x = () ⇒ document.write("hi all")

x() // function call

② Var x = _ ⇒ document.write("hill 123")

x() // function call

③ Var x = name ⇒ document.write("student name is" + name + "
")

x("sai") // function call

④ let x = Sub → { return Sub }

var y = x("Angular js") (or) document.write(x("Angular js"))

document.write(y); // function call

⑤ let x = Sub → Sub

document.write(x("react js")) // function call

End functionality part-

25/03/23

33 inbuilt

String:-

String methods & properties:-

1) JavaScript string Length:-

The length property returns the length of a string.

Eg:- let txt = "ABCD Sai Kumar Reddy";
let length = txt.length;
Console.log(length) // 17

Note:- It is not a function it is a property. In java it is method.

2) slice():

Syntax:- slice(start, end)

slice() extract a part of a string. and returns the extracted part in a new string.

The method takes 2 parameters: the start position, and the end position (end not included) .. end index value not will print.

Eg:-

let str = "Apple, Banana, Kiwi";
let part = str.slice(7, 13);
Console.log(part); // Banana

Slice Accept Negative indexes:-

If a parameter is negative, the position is counted from the end of the string.

This Example slices out a Portion of a string from position -12 to position -6:

Eg:-

let str = "Apple, Banana, Kiwi";
let part = str.slice(-12, -6);
Console.log(part); // Banana

3) substring():

Syntax:- substring(start, end)

Substring() is similar to slice().

The difference is that substring cannot Accept Negative indexes.

Eg:-

```
let str = "Sai Kumar Reddy";
let part = str.substring(5, 14);
Console.log(part); // umar Redd
```

(4) Substr():-

Syntax:- substr(start, length);

Substr is similar to slice().

The difference is that the second parameter specifies the length of the extracted part.

Eg:-

```
let str = "Apple, Banana, kiwi";
let part = str.substr(6, 6);
console.log(part); // Banana
```

(5) replace():

The replace() method replaces a specified value with another value in a string:

By default, the replace() method replaces only the first match.

Eg:-

```
let text = "please visit microsoft!";
let newtext = text.replace("microsoft", "spider");
Console.log(newtext); // please visit spider.
```

Note:- It will always replace the first match string.

(6) trim():-

The trim() method removes white spaces from both sides of a string:

Eg:-

```
let text1 = "   HELLO World!   ";
let text2 = text1.trim();
```

→

document.write("Length text1 =" + text1.length + "
");
document.write("Length text2 =" + text2.length + "
");

Output :-

Length text1 = 14

Length text2 = 12

Note :- Trim() function is remove spaces in end and begining of the string.

7 ⑦ String indexOf() :-

String indexOf() :

The indexOf() method returns the index of (the position of) the first occurrence of a specified text in a string.

indexOf() returns -1 if the text is not found.

Eg:- let str = "please locate where 'locate' occurs";

str.indexOf("locate"); // 7

str.indexOf("kocate"); // -1

str.indexOf("ate"); // 10

str.indexOf("late"); // -1

str.indexOf("e"); // 2

⑧ String lastIndexOf() :-

The lastIndexOf() method returns the index of the last occurrence of a specified text in a string.

lastIndexOf() returns -1 if the text is Not found.

Eg:-

let str = "please locate where 'locate' occurs";

str.lastIndexOf("locate"); // 21

str.lastIndexOf("kocate"); // -1

str.lastIndexOf("ate"); // 24

str.lastIndexOf("late"); // -1

str.lastIndexOf("e"); // 26

str.lastIndexOf("s"); // 34

Note :- A character is present many times at that time we want to last occurence character at that time we goto lastIndexOf().

(9) String includes():

Syntax:- string.includes(Searchvalue, start)

Searchvalue:- Required. The string to search for given string.

Start :- optional. Default 0. position to start the search.

returns:- Returns true if the string contains the value, otherwise false.

The includes() method returns true if a string contains a specified value.

Eg:- let text = "Hello world, welcome to the universe.";
text.includes("world"); // True
text.includes("world", 7); // false // will start index value from 7 → orld - false

(11) repeat(): like for loop:-

It will repeat specified string based on the number we pass.

Eg:-
var str = "java Script";
var x = str.repeat(5);
console.log(x);

o/p
java Script
java Script
java Script
java Script
java Script

(12) charAt():

It will return the character of the specified position

Eg:- var str = "hi guys how are you all";
var x = str.charAt(3);
console.log(x); // g

(13) charCodeAt():

It will return the assci value of specified index value of that character.

Eg:- var str = "javaScript";
var a = str.charCodeAt(1);
Console.log(a); // 97

Non primitive datatypes 27/03/23

Array :-
Collection of heterogeneous data.

Syntax to declare a array :-

① Var ArrayName = [data1, data2, data3 ... datan];

② var ArrayName = new Array();

③ Var ArrayName = new Array(data1, data2, ... datan);

In java script we can represent Array in 3 ways :-
i) Array literal concept.
ii) using new keyword.
iii) using Constructor

Ex :- 1
```
<script>
    Var object = [12, "Sai", true, null, 76.66, "Sai"]
    Console.log(object)
    document.write(object)
</script>
```

if you want store the data in the form of Table the data which is present in Array.

```
<script>
    var object = [12, "Sai", true, null, 76.66, "Sai"];
    Console.log(object)
    document.write(object)
    Console.table(object)
</script>
```

O/P console

index	value
1	Sai
5	Sai
0	12
4	76.66
2	true
3	null

* Note :- Console is a API inside Console we have so many inbuilt functions.

Ex:-2
```
<Script>
    var x = new Array();
    x[0] = "js";
    x[1] = "java";
    x[2] = 85687;
    x[3] = "html";
    x[100] = "node js";
Console.log(x); // (101) ['js', 'java', 85687, 'html', empty x96, 'node js']
Console.log(x[2]); // 85687;
Console.log(x[67]); // undefined
Console.log(x[1000]); // undefined
</Script>
```

Ex:-3
```
<script>
    var num = new Array("js", 56, "node js");
    Console.log(num); // B ['js', 56, 'node js']
</script>
```

Call Back function:- → call back function

function passing as an argument we call it as call back function.

Ex:-
```
function ABC(para)
{
    logic
}
ABC( () => "js" ); // function call
```

```
function xy = ()
{
    return "js";
}
```

(*) if you want find ur current position using javaScript ?

<script>
var successCallback = position => console.log(position)
var errorCallback = position => console.log(position)
navigator.geolocation.getCurrentPosition(successCallback, errorCallback)
</script>

---

* javaScript Array methods :- 28/03/23

① Concat():

It returns a new Array object that contains two or more merged arrays

<script>
Ex:- var x=[10,30,40,50]
var y=[11,12,13,14,15,16]
var y1=[11,12,13,12,13,11]
document.write(x.concat(y,y1)+"
");
document.write(x+"
") [10,20,40,50]
document.write(x.concat(y)+"
") [10,30,40,50,11,12,13,14,15,16]
</script>

merged one

more than one merged arrays

[10,30,40,50,11,12,13,14,15,16,11,12,13,12,13,11]

② push():

It adds one or more elements to the end of an array.

Ex:- <script>
var x=[10,20,30,40]
document.write(x+"
")
document.write("after push() function"+"
")
x.push(50,60,70)
document.write(x);
</script>

o/p

10 20 30,40
after push() function
10,20,30,40,50,60,70

6 & 9

5 & 8

④ Unshift() :- unshift()
It adds one or more elements in the begining of the given array.

Ex :- <script>
var x=[10,20,30,40]
document.write(x). [10,20,30,40]
document.write("after unshift() function"+"
") // - - -
x.unshift(50,60,70) [50,60,70,10,20,30,40]
document.write(x);
</script>

⑤ POP() :-
It removes and returns the last element of an arrays.

Ex :- <script>
var x=[10,20,30,40]
document.write(x+"
") // [10,20,30,40]
document.write("Poped data is" + x.pop() + "
") // ...40
document.write(x); // [10,20,30]
</script>

⑧ Shift() :-
It removes and returns the first element of an array.

Ex :- <script>
var x=[10,20,30,40]
document.write(x+"
") // [10,20,30,40]
document.write("~~Poped~~ data is" + x.shift() + "
") // ...10
document.write(x); // [20,30,40] (shifted)
</script>

⑦ reverse() :
It reverse the elements of given array.

Ex :- <script>
var x=[10,20,30,40]
d.w(x.reverse()+"
") // [40,30,20,10]
</script>

(2) Sort():-

It returns the element of the given array in a Sorted order.

Ex:- <script>

Var x=[30,120,10,200,5];

1 document.write(x+"
") // [30,120,10,200,5]

2 document.write(x.sort()+"
"); // [5,10,30,120,200] Ascending order

</script>

if you want descending order

first perform sort() than perform reverse():

3 document.write(x.reverse()+"
") // [200,120,30,10,5]

(or)

* (2,3) In one line :- document.write(x.sort().reverse()) // Descending

---

=> (1) WAJS program to print biggest element in the array?

Var x=[20,60,30,10,40]

document.write(x.sort()+"
") [10,20,30,40,60]

document.write(x.pop()+"
") // 60

In one line :- document.write(x.sort().pop())

=> (2) WAJS program to print lowest element in the array?

Var x=[20,60,30,10,40]

document.write(x.sort().shift()+"
") // 10

(3) WAJS code to reverse given string ?

Var x="banglore" o/p

Console.log(str) // banglore

Var arraystr=str.split("") ['b','a','n','g','l','o','r','e']

Console.log(arraystr)

Var rev=arraystr.reverse()

Console.log(rev) ['e','r','o','l','g','n','a','b']

Var result=rev.join("")

Console.log(result) // erolgnab

split() → It is used to split the string into characters (string function)
join() → join is used to merge or convert characters into string (string function)

PAGE EDGE
Date : / /

(10) write code to reverse the string in single line?

```
var x = "banglore";
document.write(x.split("").reverse().join("") + "<br>");
```

Continue the array methods 29/03/23

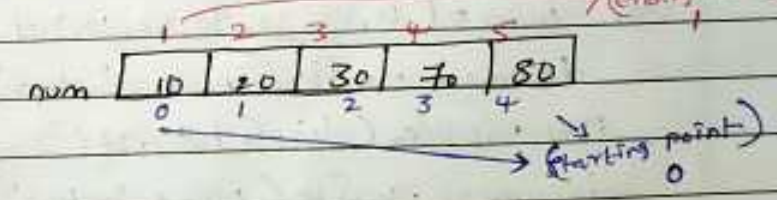
(11) slice():-

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

Syntax:- slice(starting point, ending point)

Ex:-

```
<script>
var num = [10,20,30,70,80]
document.write(num+"<br>") // [10,20,30,70,80]
document.write(num.slice(0,5)+"<br>") // [10,20,30,70,80]
document.write(num.slice(3,5)+"<br>") // [70,80]
document.write(num.slice(3,4)+"<br>") // [70]
document.write(num+"<br>"); // [10,20,30,70,80]
</script>
```



(12) splice():-

It add/remove elements to/from the given array.

Syntax:- splice(starting point, delete count, [add element])

Ex:- (i) Using splice() to removed the data

```
var num = [10,20,30,70,80]
document.write(num+"<br>") // [10,20,30,70,80
num.splice(2,2);
document.write(num+"<br>"); // [10,20,80]
num.splice(0,1) // starting point only
document.write(num); // [20,30,70,80]
num.splice(4,1)
document.write(num); // [10,20,30,70]
```

Explanation:- num.splice(2, 0, 50, 60) → adding elements from 2nd index

2 → starting index; 0 → how many elements you want to delete

Ex:- using Splice() we can add the elements to the array of anywhere

```
Var num=[10,20,30,70,80]
document.write(3,0,50,60)
document.write(num+"<br>"); [10,20,30,70,80]
```

* num.Splice(3, 0, 50, 60) — start index; 0 → delete elements; adding elements.

```
document.write(num+"<br>"); [10,20,30,50,60,70,80]
```

* num.Splice(start 0, 0, 21, 22, 23)
* num.Splice(start 7, 0, 75, 76, 77) — adding elements

o/p

num = [10, 20, 21, 22, 23, 30, 70, 75, 76, 77, 80]

0 1 2 3 4 5 6 7 8 9 10

** *** if you want to perform both add and remove at one statement

* must have the adding & deleting index value is same than only we can perform both at a time.

```
Var num=[10,20,30,70,80,10,20]
document.write(num+"<br>") [10,20,30,70,80,10,20]
num.Splice(1,2,67,89,76)   deleting  adding
document.write(num+"<br>"); [10,67,89,76,70,80,10,20]
```

** (Q) what are the differences b/w Slice() and Splice() functions

Slice()	Splice()

until it encounter No value.

③ filter() :- higher order function (call back function).

It returns the new Array Containing the elements that pass the provided function conditions.

Syntax :-

Ex :-

```
var num = [10,20,30,70,80,10,20]
```

F F F T T F F

10>30 - F
20>30 - F
30>30 - F
70>30 - T
80>30 - T
10>30 → F
20>30 → F

n=10 % 70 80

```
var result = num.filter(n => n>30);
```

arrow function

```
function x(n)
{
  return n>30
}
```

```
document.write(result); // 70, 80
```

④ find() :- higher order function (call back function)

It returns the value of the first Element in the given array that satisfies the specified condition.

It only Accept the first value which is true for the condition

Ex :-

```
var num = [10,20,30,70,80,10,20]
var result = num.find(n => n>30)
document.write(result); // 30
```

10>20 - F
20>20 - F
30>20 → T - stop here

Note :-

filter :- The resultant data will be stored in the form of array [].

find :- The resultant data will be stored only data.

// Completed array methods

⑤ How many ways in javascript to iterate the data in forward way?

Forward Iteration 6 ways

```
<script>
① Var num = [10,20,30,70,80,10,20]
  document.write(num);
② for(let i=0; i<num.length; i++){
    console.log(num[i])
  }
③ num.forEach((n,i,num) => Console.log(n,i,num))
④ num.map((n,i,num) => Console.log(n,i,num))
⑤ for(let key in num){
    Console.log(num[key])
  }
⑥ for(let iterator of num){
    Console.log(iterator)
  }
</script>
```

In java we have 5 ways.

30/03/23

Non primitive

window (super most object in js)

Object :-

Object is a real world physical Entity.

In javaScript we can Create an object 3 ways.

→ 1. object literal Concept.

→ 2. JSON (javaScript object Notation) format.

→ 3. using class Concept.

① using object literal :-

Syntax:- Var objectName = { key1 : "value1",
key2 : "value2",
key3 : "value3",
!
keyn : "valuen" }

Ex:- <script>

```
Var person = { name : "Sai",
               age : 22,
               Sal : "Lulpa",
               hobbies : ["sports", "singing"],
               ad : {
                    state : "Andhra",
                    area : "KKUP",
               } } city : "kadapa"

Console.log(person)
Console.log(person.Sal)
Console.log(Person.hobbies[0])
Console.log(person.add.area)
```

Assume if there is a 'n' number of lines code. if you want to add any Key Value pair you can use (ObjectName.)

```
person.email = "saimanchala72143@gmail.com";
person.add.PinCode = 516259
```

</script>

role of JS

you should to add any data to the database to fetch you should convert into JSON

② JSON format :-

Syntax :- var objectName = { "key1" : "value1",
"key2" : "value2",
!
'keyn' : "valuen"
}

* JSON (Javascript object Notation) → it is used to add the data to the database and fetch the data from the database.

* we can Convert object literal data in the form of JSON using 2 ways
* (i) with internet :- json formatter and validator → application
(ii) without internet :- using stringify() inbuilt function

stringify() :- it will Convert object literal data in the form of Json.

Ex :-
```
<script>
Var person = { name : "sai",
               hobbies : ["sports", "dance"],
               add : {
                     state : "karnataka",
                     area : "banglore"
               } }

Console.log(person)
Var result = JSON.stringify(person)
Console.log(result);
</script>
```

if you want to store multiple objects example multiple employees:
whenever you have 'n' numbers of data :- we can do with array [ ] inside two { }

```
Var emp = [ {
              name : "sai",
              sal : "6lpa",
              age : 23
            },
            {
              name : "Nayan",
              sal : "8lpa",
              age : 25
            },
            {
              name : "suraj",
              sal : "9lpa",
              age : 24
```

```
Console.log(emp)
Console.log(emp[1].sal)
</script>
```

Advance level 31-03-2023

(Fetching the data from a particular link)

Real time example

```
<script>
  Var url = "https://api.github.com/users"
  async function abc()
{
  Var response = await fetch (url) // fetch data from url
  Var data = await response.json () // convert .json to object-like form
  Console.log (data) // print

  for (let i=0; i< data.length ; i++)
  { // display the one by one object
    document.write (data[i] + "<br>")
    // display avatar-url link one by one
    document.write (data[i].avatar_url + "<br>")
  }}
  abc() // function call
<\script>
```

Note:- Converting Json to object literal we make use of '.json()'

Async wait Function :-

A Asynchronous is used to achieve multithreading Concept. multithreading means performing multiple tasks simultaneously or at a time.

:Explaination:-

```
async function Abc ()
{ var Response = await fetch ("URL")
}
Abc ()// f-c
```


Note:- await- is give permission to fetch() function fetch the data from what I gave the 'URL' and give data in the form response & that data will be in 'Json format' convert .json()

This part of Variable Concept

PAGE EDGE
Date : / /

① Variable Hoisting :-

moving Variable declaration part at the top of original code or Native code this process we call it as Variable Hoisting

Scenario-1

D.V.

```
<script>
document.write(x);
</script>
```

J.E.View

```
<script>
document.write(x);
</script>
```

memory Allocation

Web browser

x is Not defined

Scenario-2

D.V.

```
<script>
document.write(x);
var x=900;
</script>
```

Variable Hoisting

J.E.V.

```
<script>
var x;
document.write(x);
var x=900;
</script>
```

M.A

① x;
② undefined
③ 900

Web browser

undefined

Ex:-

```
<script>
var h="hi"
document.write(d+"<br>") // undefined
var t="java Script"
var a=true;
var b="bye";
document.write(a+"<br>") // true
document.write(s+"<br>") // undefined
var w=789
document.write(h+"<br>") // bye
var d=345
document.write(w+"<br>") // 789
document.write(t+"<br>") // javaScript
var s=56.789.
</script>
```

Java Script View :-

```
<script>
var h;
var t;
var a;
var w;
var d;
var s;
</script>
```

M.A

h = undefined "hi" "bye"
t = undefined "javaScript"
a = undefined true
w = undefined 789
d = undefined 345
s = undefined 56.789

W.B

undefined
true
undefined
bye
789
javaScript

Async await :- The await keyword can only be used inside an async function. The await keyword makes the function pause the execution and wait for a resolved promise before it...

01-04-2023

③ class Concept :-

The class is a blueprint of object.

Syntax :- class classname {

}

using class Concept Creating object :-

Syntax :- Var objectref = new classname();

Ex :-

```
<script>
class Signature {
        price = "150"
        quan = "150 ml"
        Color = "Green"
}
→ Const S1 = new Signature();
    console.log(S1)
</script>
```

or

```
Color : "Green"
price : "150"
quan : 150 ml
```

Note :- We can't create object inside the class Scope.

We can create multiple objects for the class

```
→ Const S2 = new Signature();
→ Const S3 = new Signature();
    Consde.log(S2)
    Console.log(S3)
```

* Constructor :-

Constructor is the member of the class. it must be the inside the class. Syntax :- constructor(){ }

Ex :-

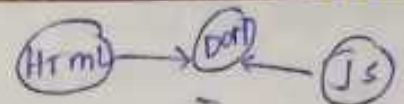
```
class lipsticks {
    price;
    brand;
    Color;
    type;
```

```
constructor(b,p,c,t){// parameterised Constructor
{->
    this.brand=b;
    this.price=p;
    this.color=c;
    this.type=t;
[document.write(this.price+" "+this.brand+" "+this.color+" "+this.type
+"<br>");
<-}
<-}
Const l1 = new lipsticks("nykaa",499,"hot pink","matt")
Const l2 = new lipsticks("Sugar",799,"babypink","regular")
Const l3 = new lipsticks("MAC",1499,"DarkPink","MATT")
Const l4 = new lipsticks("LAKME",399,"LIGHT PINK","REGULAR")
</script>
</body>
</html>
```

Constructor Calling statement :- object Creation statement Constructor invocation.
Inside the Constructor if we want to initialize or Retrieve the data - we must 'this' keyword.
'this' keyword Refers Current invoking object reference address.

Note :- * inside constructor we must use 'this' keyword to initialize the data members.
* if we want to initialize values we need not to declare the variables.

Back end is Completed



Front End

Document object Model :- (DOM)

"The Document object Model (DOM) is a platform and language-neutral interface that allows Programs and Scripts to dynamically access and update the Content, structure, and style of a document."

D Document ⟹ file
O object ⟹ tags elements
M Model ⟹ layout, Structure (Tree like structure)

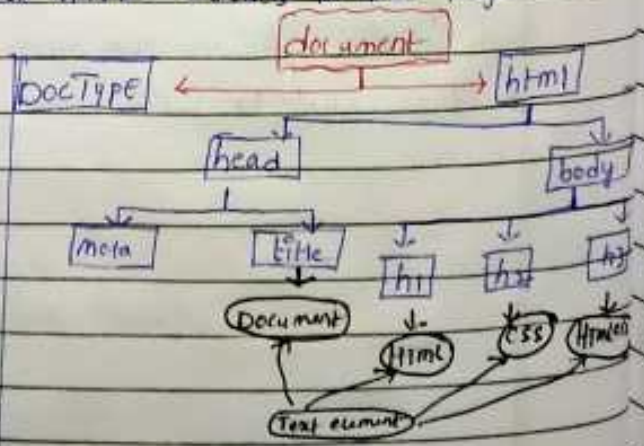
When a webpage is loaded, the browser Creates a Document object Model of the page.

The HTML DOM model is Constructed as a tree of objects:

With the object model, JavaScript gets all the power it needs to Create HTML:

JavaScript Can change all the HTML elements in the page.
JavaScript Can change all the HTML attributes in the page
JavaScript Can change all the CSS styles in the page
JavaScript Can remove existing HTML elements and attributes
JavaScript Can add new HTML elements and attributes
JavaScript Can react to all existing HTML events in the page
JavaScript Can Create new HTML events in the page.

```
<DocType>
  <html>
  <head>
    <meta>
    <title> Document </title>
  </head>
  <body>
  <h1> HTML </h1>
  <h2> CSS </h2>
  <div> HTML CSS </div>
  </body>
</html>
```



D - document - Html file
O - object - Tag
M - Model - Tree Structure

CSS Selectors :- id, class, group, universal, attribute, Element

01/04/23

PAGE EDGE
Date : / /

Document Model Selectors (DOM) Selectors :-

1) document.getElementsByTagName();
2) document.getElementById();
3) document.getElementsByClassName();
4) document.querySelector();
5) document.querySelectorAll();

03-04-2023

(1) getElementsByTagName() :-

⇒ This function is used to target or fetch and element from the document object Model. it will select multiple tags (in the form of array) this function target the element based on tagname.

Ex :-

```
<html>
<body>
  <h1> html </h1>
  <h1> CSS </h1>
  <h1> is </h1>
  <h1> react js </h1>
<button onclick="ABC()" ondblclick="doubleclick()"> red color </button>
<button onclick="clr()" > reset </button>
<script>
  function ABC(){
    var tag = document.getElementsByTagName("h1")
      → All the data stored in the form array kind of data : h1 tag
    for(let i=0; i<tag.length; i++){
      tag[i].style.color = "red"
      tg[i].style.background = "aqua"
    } }
  function clr(){
    location.reload() // inbuilt object provided by window
  }                    it is used to reload the page.
  function doubleclick(){
    var x = document.getElementsByTagName("h1");
    for(let i=0; i<x.length; i++){
      x[i].style.border = "2px solid blue"
    }}
```

03/04/23

② document.getElementById :-

=> • getElementById by id function targets and returns only one particular specified.
• element based on id value.
• id cannot have duplicate value if we have the it will target the first encounter id only.
• It is capable of handling only one id.

Ex:- <script> <body>

<h1> i) Select Scripting language </h1>
<input type="radio" name="r1" id="input1"> JAVA

<input type="radio" name="r1" id="input2"> HTML

<input type="radio" name="r1" id="input3"> CSS

<input type="radio" name="r1" id="inputy"> JS

<button onclick="valic()" value="Submit"> Submit </button>
<script>
function valic(){
var x1 = document.getElementById("input1");
var x2 = document.getElementById("input2")
var x3 = document.getElementById("input3")
var x4 = document.getElementById("inputy")
if(r1.checked == true){
alert("Answer Selected is JAVA")
else if(r2.checked == true){
alert('Answer Selected is HTML')
else if(r3.checked == true){
alert('Answer Selected CSS")
else if(r4.checked == true){
alert("Answer Selected JS");
else{
alert("No Answer Selected").
}
}
</script>

2 → Ex°. 2

```
<style>
  div{
      border : 2px solid ;
      width : 50% ;
      margin : auto ;
      text-align : center ;
      background-color : antiquewhite ;
    } padding : 10px ;
  h1{
      border : double red;
      font-style : italic ;
    } background-color : rgb
</style>
<head> <body>
<div>
    <h1 id = "heading" > </h1>
    < button onclick = "incre()" > inc </button>
    < button onclick = "decre()" > dec </button>
    < button onclick = " reset()" > reset </button>
</div>
<script>
    count = 0;
  function incre(){
    x = count + 1;
    document . getElementById ( "heading"). innerHTML = x
  }
  function decre(){
    y = count - 1;
  } document . getElementById ( "heading"). innerHTML = y
  function reset (){
      count = 0;
      document . getElementById ( "heading"). innerHTML = count
  }
</script>
```

③ getElementsByClassName():

This function targets and returns all the elements with matching class value.

It returns HTML Collection object and as to the access to using index value.

Ex:-

```
<body>
  <h1 id="blue">1. JAVA </h1>
  <h1 class="orange">2. JAVASCRIPT </h1>
  <h1 class="orange">3. SPRING </h1>
  <h1 class="orange">4. REACT </h1>
  <h1 id="blue">5. HTML </h1>
  <button onclick="show()"> orange </button>
<script>
  function show(){
    var x = document.getElementByClassName("orange")
    for(let i=0; i< x.length; i++){
      x[i].style.color = "orange"
    }
  }
</script>
```

05/04/23

PAGE EDGE
Date : / /

Project [filter data]

```
<style>
  * {
      background-color : white;
    }

  #myinput {
        width:100%;
    }
  #mytable {
        width : 100%;
        background-color: blueviolet;
      } border : 2px solid black;
    th {
        width : 500px;
        background-color : yellow;
        text-align : center;
      } color : blue;
    tr td {
        color : red;
        width : 500px;
        border : 2px solid black;
        text-align : Center;
      }

</style>
<body>
  <input type="text" id="myinput" onkeyup="display()" >
  <br> <br>
  <table id="mytable" >
    <tr class="header">
        <th> name </th>
        <th> degree </th>
        <th> profession </th>
    </tr>
```

```
<tr>
    <td> Sai </td>
    <td> Bcom </td>
    <td> front End developer </td>
</tr>
<tr>
    <td> Narayan </td>
    <td> mech </td>
    <td> Test Engineer </td>
</tr>
<tr>
    <td> Suraj </td>
    <td> mech BE </td>
    <td> back End Developer </td>
</tr>
</table>
<script>
  function display(){
  filter = document.getElementById("myinput").value.toUpperCase();
  table = document.getElementById("mytable")
  tablerow = table.getElementsByTagName('tr')
  for( let i=0; i< tablerow.length; i++)
  {
    tabledata = tablerow[i].getElemeantByTagName('td')[0]
    if(tabledata){
        data = tabledata.innerHTML
        if( data.toupperCase().indexOf(filter) > -1)
          {   tablerow[i].style.display = ""
          }
        else{
            tablerow[i].style.display = "none"
    } } } }
</tbody>
</table>
```

myinput

project is whatever type in myinput text-field

eg: SaI

In whatever that letters present that all names should be displayed.

SaI

myTable

<th> Name	<th> Degree	<th> Profession	</tr>
<td> Sai	<td> BCome	<td> front end develop	</tr>
<td> Nayan	<td> Mech	<td> Test Engineer	</tr>
Suraj <td>	<td> MechBE	<td> Back End Develop	</tr>

6-04-2023

Project (local storage)

```
<style>
  h1{
    text-align : Center;
  }
  input, button{
        padding : 10 px ;
        height : 30 px;
        background-color : white ;
    }
  fieldset {
      margin-bottom : 20 px;
</style>  </style>
<body>
  <h1> local storage - java Script </h1>
<fieldset>
  <legend> legend data </legend>
<input type = "text"  placeholder = "Enter key"  id = "input-key" >
<input type = "text" placeholder = "enter value" id = "input-value" >
<button onclick = "display()" > Submit </button>
</fieldset>

<fieldset>
    <legend> local storage </legend>
    <div id = "output" > </div>
</fieldset>
<script>
  Const input-key = document.get-Element-By Id ("input-key")
  Const input-value = document.get Element-By Id ("input-value")
  Const output = document.get Element-By Id ("output")
  ->
```

```
function display(){
    const key1 = input-key.value
    const value1 = input-value.value

    if(key1 && value1)
    {
        localstorage.setItem(key1, value1)
    } location.reload();
}
for(let i=0; i< localstorage.length; i++)
{
    const key2 = localstorage.key(i)
    const value2 = localstorage.getItem(key2)
    output.innerHTML+= " "+key2+" :"+value2+"<br>"
}
</script>
</body>
```

7/04/23

Events :-

Event is basically an action perform. for respective event handler should be attached.

Note :- events are used to call the function inside script itself.

Syntax :- Targeted Element. event = handler function ;

Ex:1

```
<body>
  <button id="btn" > click me </button>
  <script>
    // logic
    function display(){
      alert("event function");
    }
    // event
    var x = document.getElementById("btn")
    x.onclick = display
  </script>
</body>
```

events → onclick

using arrow function

Ex:2

```
<body>
<button id="btn" > click me </button>
<script>
  var x = document.getElementById("btn")
  x.onclick = () => alert("event function")
</script>
</body>
```

event drawback :- With using event we can only invoke only one function. If u write multiple functions it will invoke last function

Ex :- →

```
<body>
  <button id="btn" > payment </button>
<script>
    var x = document.getElementById("btn")
x.onclick = () => alert("payment through phonepay")
x.onclick = () => alert("payment through Google pay")
x.onclick = () => alert("payment through Amazon pay")
</script>
</body>
```

o/p
payment through Amazon pay

(*) To overcome this drawback presented in the event we can use the concept of addEventListener(): ✓

overwriting the event handler function:-

whenever we attach more than one event handler per one particular event, then the lastly (latestly) attached event-handler will be considered.

Syntax:- addEventListener("Event", handlerfunction, [Boolean])

Ex:-

```
<body>
  <button id="btn" > payment </button>
<script>
    var x = document.getElementById("btn")
// addEventListener("Event", function, [boolean])
x.addEventListener("click", () => alert("amazon pay"))
x.addEventListener("click", () => alert("phone pay"))
x.addEventListener("click", () => alert("google pay"));

</script>
</body>
```

Actual codes

```
function ap(){
  alert("Amazon pay");
}
```