Array : Array is collection of elements. Each element is combination of index and value. Javascript is loosely typed script so we can store any type of values in array.

```
<script>
 var arr=[10,20,30,40,50,"scott"];
 alert(arr[5]);
 alert(arr);
</script>
```

push : using this function we can add an element at the end of arrya and it returns total number of elements.

```
<script>
 var arr=[10,20,30,40,50];
 var rv=arr.push(60);
 alert(arr);
 alert(rv);
</script>
```

pop : To remove the last element of array and returns value of that element.

Ex:

```
<script>
 var arr=[10,20,30,40,50];
 var rv=arr.pop();
 alert(arr);
 alert(rv);
</script>
```

shift : To remove the first element of array and returns value of that element.

```
<script>
 var arr=[10,20,30,40,50];
 var rv=arr.shift();
 alert(arr);
 alert(rv);
</script>
```

unshift() : Adds an element at the begining of array and returns total number of elements.

```
<script>
 var arr=[10,20,30,40,50];
 var rv=arr.unshift(2);
 alert(arr);
 alert(rv);
</script>
```

---

slice() : To get some part of array.

```
<script>
 var arr=[10,20,30,40,50];
    var narr=arr.slice(2,4);
    alert(narr);
</script>
```

---

splice() : To add/remove elements based on the index number.

```
<script>
 var arr=[10,20,30,40,50];
 arr.splice(1,0,111,222,333,444,'abc');
 alert(arr);
</script>
```

```
----------------------------
for..in : Runs a loop through the elements of array and returns index of those
elements.

Ex:
-------------------------
<script>
 var arr=[10,20,30,40,50];
 for(var x in arr){
     alert(arr[x]);
     }
</script>

-----------------------
for..of : Same as for..in but returns values of elements.

<script>
 var arr=[10,20,30,40,50];
 for(var x of arr){
     alert(x);
     }
</script>

---------------------
forEach() : It is combination of previous two functions holds both index and values. It
also executes a function for every iteration.

Ex:

<script>
  var arr=[10,20,30,40,50];
  arr.forEach(function(val,ind){
    console.log(ind,val);
  })
</script>



-----------------------
map() : It is same as forEach to run a loop through the elements of array. Map returns
values. Total number of return values are equal to total number of elements.

<script>
  var arr=[10,20,30,40,50];
  var narr=arr.map(function(val,ind){
    console.log(ind,val);
    return val+ind;
  })
  console.log(narr);
</script>

-----------------------
filter() : Runs a loop through the elements of array and returns the truth values.

<script>
  var arr=[10,20,30,40,50];
  var narr=arr.map(function(val,ind){
    return val>30;
  })
  console.log(narr);
</script>

----------------------

split() : To split a string as array elements based on input value.

<script>
  var str="welcomescott";
  var narr=str.split("o");
```

```
  alert(narr[2])
</script>
----------------------

join() : To  join the elements of array as string based on input value.

<script>
  var arr=["scott","amith","suresh"];
  var str=arr.join("/");
  console.log(str);
</script>

--------------------

Object : Object is collection of properties. Each property is combination of name and
value.

<script>
  var obj={uname:"scott",city:"hyderabad"};
  console.log(obj.uname);
</script>

Ex2 :

<script>
  var obj=[{uname:"scott",city:"hyderabad"},{uname:"john",city:"chennai"}];
  console.log(obj[1].uname);
  console.log(obj[1].city);
</script>


Ex3:


<script>
  var obj={uname:"scott",city:"hyd",stt:"Tg"};
  for(var x in obj){
    console.log(obj[x]);
  }
</script>

Object.keys() : To get all the keys of an object in the form of array

<script>
  var obj={uname:"scott",city:"hyd"}
  var keys=Object.keys(obj);
  console.log(keys);
</script>

Object.values() : To get all the values of an object as array

<script>
  var obj={uname:"scott",city:"hyd"}
  var values=Object.values(obj);
  console.log(values);
</script>

--
Object.hasOwnProperty() : To check whether the specified property is available or not in
an object.

<script>
var obj={uname:"scott",city:"hyd"};
alert(obj.hasOwnProperty("state"));
</script>

-------------------------------------------------
Remove duplicate elements from Array

```

Ex 1:

```
<script>
var arr=[10,20,30,10,40];
var obj={};
for(var i=0;i<arr.length;i++){
    obj[arr[i]]=1;
    }
console.log(obj);
var keys=Object.keys(obj);
console.log(keys);
</script>
```

Ex 2:

```
<script>
var arr=[10,20,30,10,40];
var narr=new Set(arr);
console.log(narr);
</script>
```

---

uname-scott
wife - name - sw1
child : sons - sw1c1,ss1c2 dau - sw1d1,sw1d2

name - sw2
child : sons - sw2s1 dau - sw2d1

Ex:

```
<script>
var ob={uname:"scott",wives:[{
    name:"sw1",child:{sons:["sw1c1","sw1c2"],dau:["sw1d1","sw1d2"]}
},{
    name:"sw2",child:{sons:["sw2s1"],dau:["sw2d1"]}
}]}
console.log(ob);
alert(ob.wives[0].child.sons[1]);
</script>
```

---

cmp - Hyd - prog - hp1,hp2 admin - ha1,ha2
bang - prog - bp1,bp2 hr - bhr1,bhr2

Ex:

```
<script>
var cmp = {hyd:{prog:["hp1","hp2"],admin:["ha1","ha2"]},
            bang:{prog:["bp1","bp2"],hr:["bhr1","bhr2"]}}
    alert(cmp.hyd.prog[0]);
</script>
```

Microsoft Visual Studio Code : It is an open source IDE to work with react, angular, node,... technologies.

---

variable : variable is name of memory location to store some values. JavaScript is loosely typed script so no need to provide data types.

Types :

Local Variable : A variable declaration inside the function we can call as local variable.

Ex:

```
<script>
    function fun1(){
        var x=100;
        alert(x)
    }
    function fun2(){
        alert("From fun2")
        // alert(x) Local variable
    }
    fun1();
    fun2();
</script>
```

Global Variable : Variable declaration outside all functions comes under global variable. We can access global variable from any function with in the page.

```
<script>
    var x=100;
    function fun1(){
        alert(x);
    }
    function fun2(){
        alert(x);
    }
    fun1();
    fun2();
</script>
```

Block scope variable : Variable declaration inside the block using let key word. A block scope variable we can not access from another block. Let introduced with ES- 6.ECMA script provides standards to JavaScript.

Ex:

```
<script>
    function fun1(){
        {
        var x=100;
        let y=200;
        }
        alert(x);
        alert(y); // Can not access because of block scope
    }
    fun1();
</script>
```

Lexical scope : It is scope of outer function of inner function.

```
<script>
```

```
    function funouter(){
        var x=100;
        alert("From outer")
        function funinner(){
            alert("From inner")
            alert(x); // calling lexical scope variable
        }
        funinner();
    }
    funouter();
</script>


Function : Function is set of executable statements to perform a task. we can use
functions for reusable. using function keyword we can  create
functions.


<script>
function fun1(){
  console.log("Function exec...");
}
fun1();
</script>

Types of functions :
1) Anonymous function : If we create a function with out any name comes under anonymous
function.

<script>
var x=function(){
    console.log("Func exec..");
}
//alert(x);
x();
</script>

2) Nested Function: It is a concept of declaration of a function inside another function.

Ex:

<script>
function fun1(){
  alert("From fun1");
    function funinner(){
    alert("From inner");
    }

    funinner();
}
fun1();
</script>

--------------------------------

Ex: 2
<script>
function fun1(){
  alert("From fun1");
    return function funinner(){
    alert("From inner");
    }
}
var rv=fun1();
//alert(rv);
rv();
</script>

------------------
```

Callback function : If we pass a function as argument of another function comes under callback.

```
<script>
function fun1(x){
alert("From fun1");
//alert(x);
x();
}
function fun2(){
alert("From fun2");
}
function fun3(){
alert("From fun3");
}
fun1(fun2);
fun1(fun3);
</script>
```

```
<script>
function calc(no1,no2,funref){
    funref(no1,no2);
}
function add(x,y){
alert(x+y);
}
function mul(x,y){
alert(x*y);
}
calc(10,20,add)
calc(10,20,mul);
</script>
```

Arrow Function : It is shortcut of function declaration and also we can use arrow function as
nested function inside the class.

```
<script>
var arr=[10,20,30,40,50];
var narr=arr.map((x,y)=> x*222)
alert(narr);
</script>
```

IIFE(Immediate Invoking Function Expression):

```
<script>
var count=0;
var increment=(function(){
var count=0;
 return function(){
    count=count+1;
    alert(count);
    }
})()
function fun2(){
    count=222;
    alert(count);
}
</script>
<body>
  <input type="button" value="Click" onclick="increment()" />
  <input type="button" value="Click" onclick="fun2()" />
</body>
```

Event : Event is an action which we are performing on html control.

click , mouseover, mouseout,....

Every event provides attributes to call the functions

click - onclick
mouseover - onmouseover
mouseout - onmouseout

this : In javascript this is an object refers current control.

Ex:

```
<body>
  <input type="button" value="Click" onclick="alert('scott');" />
  <img src="orange.jpg" width="100px" height="100px"
   onclick="this.src='all.jpg'" onmouseover="this.src='pineapple.jpg'"
   onmouseout="this.src='orange.jpg'"/>

   <input type="button" value="Click Me"
  onmouseover="this.value='Dont click me';this.type='text'"
  onmouseout="value='Please click me' "/>
  <input type="button" value="Click" onclick="fun1(this)" />
</body>
<script>
function fun1(t){
t.value='Login'
t.type='text'
}
</script>
```

---

id : Using this property we can provide identity to a control. id we can use to call a control
from another control.

```
<body>
  <img src="orange.jpg" width="100px" height="100px" onclick="i1.src='all.jpg'"/>
  <img src="pineapple.jpg" width="100px" height="100px" id="i1"/>
</body>
```

Ex:2

```
<body>
<img src="all.jpg" width="600px" height="500px" id="imgbig"  />
<br />
<img src="orange.jpg" width="200px" height="200px" onclick="imgbig.src=this.src"/>
<img src="all.jpg" width="200px" height="200px" onclick="imgbig.src=this.src"/>
<img src="pineapple.jpg" width="200px" height="200px" onclick="imgbig.src=this.src"/>
</body>
```

Hoisting : It is a concept of declaration of variables before starting the execution of script.
var contains undefined and let does not have any value at the time of hoisting. If we are trying to access
let before initialization it gives reference error and stops the execution of script.

functions can also hoist.

```
<script>
fun1();
function fun1(){
alert("From function");
}
</script>
```

Note : Arrow functions can not be hoist

Ex:

```
<script>
fun1();
fun1=()=>{
alert("From arrow");
}
</script>
```

Object : object is collection of properties. Each property is combination of name and value.

There are different ways to create object

1) Object Literals : using {} we can create the object.

```
<script>
var obj={user:"scott",admin:"amith"};
console.log(obj.user);
console.log(obj.admin);
</script>
```

2) Using Object.create() function

```
<script>
 var obj=Object.create({
    uname:"john",
    city:"Mumbai"
    })
//console.log(obj);
console.log(obj.uname);
</script>
```

3) Using new keyword

```
<script>
var obj=new Object({
    uname:"Rajesh",
    wife:"rw"
})
console.log(obj.uname);
</script>
```

4) Using class

Object Destructor : It is a concept of creating the variables with the names of object properties.

Ex:

```
<script>
var obj={uname:"scott",city:"hyd"}
var {city,uname}=obj;
console.log(uname);
console.log(city);
</script>
```

Array Destructor : It is concepts of creating variables with the values of array.

```
<script>
var arr=[10,20,30,40,50];
var[x,y,z,a,b]=arr;
alert(x);
alert(y);
alert(z);
```

```
alert(a);
alert(b);
</script>

------------------------------

BOM(Browser Object Modal); Every browser provides some objects those are window, document, navigator, ...

window is primary object provides many properties and methods along with other objects like
document, navigator,....

alert() : using this method we can display message box on browser.

<script>
window.alert("Hi scott");
</script>

prompt() : To display input dialog box.

<script>
var rv=window.prompt("Enter name");
alert(rv);
</script>

confirm() : To display confirmation dialog box
<script>
var rv=confirm("You want to close?")
alert(rv);
</script>

print() : To display print properties dialog box.

<script>
window.print();
</script>
<body>
  <img src="pineapple.jpg" width="100px" />
</body>

Ex:

<script>
function funprint(){
  var cnf=window.confirm("You want to print ?");
    if(cnf){
    window.print();
    }
}
</script>
<body>
 <input type="button" value="Print" onclick="funprint()"/>
</body>

----------------------------------------

location : Using this property we can navigate from one web page to another web page.

Ex:

<script>
function fun1(){
    location="http://google.com";
}
</script>
<body>
    <input type="button" value="Click" onclick="fun1()" />
</body>
```

```
------------------------------------
open() : Using this method we can open another web page in current tab/ new tab/ new window.

<script>
function fun1(){
   // window.open("http://google.com","_self");
   window.open("http://fb.com","_blank");
   window.open("http://google.com","_blank","width=600,height=400")
}
</script>
<body>
    <input type="button" value="Click" onclick="fun1()" />
</body>

----------------------------
setTimeout() : To execute set of statements after specified time. Arguments are a function/set of statements and time in milli seconds

<script>
    function fun1(){
        alert("Function called")
    }
    window.setTimeout("fun1()",5000);
</script>

------------------------------
setInteral() : It is same as setTimeout()  but executes a function for every regular intervals of time.

clearInterval() : To stop the functionality of setInterval();

<script>
    function fun1(){
        alert("Function called")
    }
   var timer= window.setInterval("fun1()",5000);
</script>
<body>
    <input type="button" value="Click" onclick="clearInterval(timer)" />
</body>

--------------------------------

document : using this object we can work with current document. document is an object available inside the window object.
title : To get/set the title of document.

<script>
    function fun1(){
       document.title="My New Site"
       alert(window.document.title)
    }
</script>
<body>
    <input type="button" value="Click" onclick="fun1()" />
</body>

---------------------------------
document.URL : To get the current document url address.

<script>
    function fun1(){
       alert(document.URL)
    }
</script>
<body>
    <input type="button" value="Click" onclick="fun1()" />
```

```
</body>

----------------------

document.write() : to write some content on current document.

<script>
    function fun1(){
        document.write("Welcome")
    }
</script>
<body>
    <input type="button" value="Click" onclick="fun1()" />
</body>

----------------
document.getElementById() : To get an element from the current document based on id of
element.

<script>
    function fun1(){
        var con1=document.getElementById("t1");
        alert(con1.type);
        alert(con1.id);
        alert(con1.value)
        con1.value="ABCD";
        var con2=document.getElementById("t2")
        con2.value="Amith";
        con2.type="button";
        con2.onclick=function(){
            alert("Button clicked")
        }
    }
</script>
<body>
    <input type="text" id="t1"/>
    <br />
    <input type="text" id="t2" />
    <br />
    <input type="button" value="Click" onclick="fun1()" />
</body>
--------------------
03-02
-----
<script>
function funcalc(op){
  var txt1=document.getElementById("t1").value;
  var txt2=document.getElementById("t2").value;
if(op=="mul")
  document.getElementById("res").value=txt1*txt2;
else if(op=="sub")
  document.getElementById("res").value=txt1-txt2;
else if(op=="div")
  document.getElementById("res").value=txt1/txt2;
else if(op=="add")
  document.getElementById("res").value=parseInt(txt1)+parseInt(txt2);
}
</script>
<body>
<input type="text" id="t1" placeholder="No1"/>
<br />
<input type="text" id="t2" placeholder="No2"/>
<br />
<input type="text" id="res" placeholder="Result"/>
<br />
<input type="button" value=" * " onclick="funcalc('mul')"/>
<input type="button" value=" / " onclick="funcalc('div')"/>
<input type="button" value=" - " onclick="funcalc('sub')"/>
<input type="button" value=" + " onclick="funcalc('add')"/>
```

```
</body>

---------------------------------------
navigator : Using this object we can get the information of browser.
<script>
console.log(navigator.appVersion);
</script>

-----------------------
history : To get the history of browser.

back() : To go to the back page of browser.
go() : To go to the specific page of browser.
forward() : To go to the forward page of browser.

<body>
<h2>This is p2</h2>
<a href="p3.html">Open P3</a>
<br />
<input type="button" value="Previous" onclick="history.back()"/>
<input type="button" value="Next" onclick="history.forward()"/>
<input type='button' value='Reload' onclick="history.go(0)"/>
</body>


--------------------------
Ex:

<script>
function funswap(){
    var tmp=document.getElementById("txt1").value;
    document.getElementById("txt1").value=document.getElementById("txt2").value;
    document.getElementById("txt2").value=tmp;
}
</script>
<body>
  <input type="text" id="txt1" />
  <br />
  <input type="text" id="txt2" />
  <br />
  <input type="button" value="Swap" onclick="funswap()"/>
</body>

-----------------------------------

screen : Using this object we can get the information of current screen.

availHeight : Returns the height of the screen. It excludes scroll bar

availWidth : Returns the width of the screen.It excludes task bar

height : Returns height of screen. Includes task bar

width: returns the width of screen

<script>
alert(screen.availWidth)
alert(screen.width);
</script>

------------------------------
innerText : To set/get text on html controls
innerHTML : To get/set html content on controls.
<script>
function fun1(){
//document.getElementById("div1").innerText="Rajesh"
document.getElementById("div1").innerHTML="<img src='orange.jpg' width='50px' />"
//document.getElementById("div1").innerText="<img src='orange.jpg' width='50px' />"
```

```
}
</script>
<div id="div1">Scott</div>
<input type="button" value="Click" onclick="fun1()" />

------------
05-02

Date : Using this object we can get client system current date and time information.
It provides many methods like
getHours(), getMinutes(), getSeconds(),getDate(), getMonth(), getYear()...

Ex:

<script>
var dt=new Date();
alert(dt.getDay());
alert(dt.getHours())
alert(dt.getMinutes())
alert(dt.getSeconds())
alert(dt.getDate())
alert(dt.getMonth())
alert(dt.getYear())
alert(dt.getFullYear())
</script>

----------------

Ex:

<script>
function fun1()
{
    var dt=new Date()
    var str=dt.getHours()+ " : "+dt.getMinutes()+" : "+dt.getSeconds();
    document.getElementById("div1").innerText=str;
}
setInterval("fun1()",1000);
</script>
<body onload="fun1()">
  <div id="div1"></div>
</body>

------------------

Math : using this object we can get mathematical related information.

Math.random() : using this function we can get the random value between 0 and 1

<script>
alert(Math.random())
</script>

Math.round() : rounds a float value to its nearest integer value
<script>
alert(Math.round(10.50))
</script>

Math.floor() : Rounds a float value to its nearest lowest integer value.
Math.ceil() : Rounds a float value to its nearest highest integer value

<script>
alert(Math.ceil(10.88))
</script>

Math.sin() : to get mathematical sin value
<script>
alert(Math.sin(0))
```

```
</script>
------------------------

string : string is collection of characters. Every character contains index number.

<script>
var str="welcome";
alert(str);
alert(str[0]);
</script>

length: using this property we can get total number of characters in a string.
<script>
var str="welcome";
alert(str.length);
</script>

toLowerCase() : To convert the characters of a string into lowercase.

toUpperCase() : To convert the characters of a string into uppercase.

<script>
var str="Hello";
alert(str.toLowerCase());
alert(str.toUpperCase());
</script>

-----------------------------------------------

indexOf() : to get the index number of a character from a string

<script>
var str="welcomeoshshs jdjdjojdjdjojkjfjfjokdkd";
alert(str.indexOf("x",5));
</script>

---------------------------------------
charAt() : To get the character of specified index number

<script>
var str="welcomeoshshs";
alert(str.charAt(1));
</script>

--------------------------
split() : To split a string as array based on input character(s)
<script>
var str="welcomescott";
var arr=str.split("o")
alert(arr[1]);
</script>

--------------------------
join() : To join all the elements of array as string based on input value

<script>
var arr=[10,20,30,40,50];
var str=arr.join("");
alert(str);
</script>

----------------------

06-02
-----
charCodeAt() : to get ASCII value of input character. ASCII is a value of input key

A- 65
B -66
```

C - 67
Z-90

a-97
b-98
z-122

space - 32
enter - 13
bksp=8
tab - 9

0-48
1-49
9-57

Ex:

```
<script>
var str="abcd";
alert(str.charCodeAt(0));
</script>
```

substr() : To get the substring of input string. total return characters are same as second argument

substring() : It is same as substr but total return characters are equal to second argument-first argument.

```
<script>
var str="welcome scott";
alert(str.substring(2,5))
</script>
```

---
replace() : To replace some part of a string with new string. It replaces the first occurrences.

replaceAll() : Same as replace but replaces all occurrences

```
<script>
var str="welcome scott john scott";
alert(str.replaceAll("scott","amith"));
</script>
```

---
conversion functions : using these functions we can convert the data from one format to another format.

parseInt() : using this function we can convert input value as integer value

ex:

```
<script>
var x=100;
var y="200";
alert(x+parseInt(y));
</script>
```

parseFloat() : Converts input value as float value

```
<script>
var x="100.20";
alert(parseFloat(x))
</script>
```

eval() : Using this function we can evaluate a string as expression.

```
<script>
```

```
var x="100*20+50";
alert(eval(x))
</script>

isNaN() : To check the input value is number or not. It returns true if the input value is not a number.

<script>
var x=100;
alert(isNaN(x))
</script>

<script>
var x="10abcd0";
alert(isNaN(x))
</script>

------------------------------

Ajax : It is a web technology to send a request from browser to server without submit the web page.

Ajax stands for Asynchronous JavaScript and XML

Asynchronous : It is a process of sending a request from browser to server with irrespective of previous request / response

JavaScript : Using javascript we can create ajax object and send that object to server.

XML : Data between browser and server transfers in the form of xml/json.

---------------------

Ajax we can use to call server API services.

--------------

XMLHttpReqest
fetch
axios

using these APIs we can create ajax object and send that object to server
----------

We have different types of methods to send a request from browser to server

get : Send a request with out any data from browser to server.

post : send a request from browser to server with some data.

put : Same as post , we can use to execute update functionalities in server.

delete : Same as post , to work with delete functionalities in server.

------------------------------------------------------------
then() : then() is a callback executes when the ajax object execution is completed. It executes when the promise object execution is completed. we need to pass a function as arguments to execute some statements after execution of associated function.

Ex:

<script>
function fun1(){
fetch("https://restcountries.com/v3.1/all").then(function(dt){
    return dt.json();
}).then(function(data){
    data.map((oneCon)=>{
    document.getElementById("tab1").innerHTML+=`<tr>
    <td>${oneCon.name.common}
```

```
    <td>${oneCon.capital}
    <td>${oneCon.population}
    <td><img src='${oneCon.flags.png}' width='100px' height='100px' />
    `;
})
})
}
</script>
<body>
 <input type="button" value="Click" onclick="fun1()"/>
 <table border="1" id="tab1">
 </table>
</body>
```

---

promise : promise is an object to call functions asynchronously. By default javascript executes functions synchronously to execute them asynchronously we can use promise.

then() : It is a callback executes some statements when the promise object execution is completed.
we need to pass 2 functions as arguments one executes if promise returns success value another one executes if it returns failure value.

resolve() : using this function we can pass success value from promise object.
reject() : using this function we can pass failure value from promise object.

ex:

```
<script>
function fun1(){
return new Promise(function(resolve,reject){
reject("promise rejected")
})
}
function fun2(){
console.log("Fun2");
}
fun1().then(function(data){
console.log("First fun exec..")
console.log(data);
},function(err){
console.log("Sec fun exec..")
console.log(err);
});
fun2();
</script>
```

Ex:2

```
<script>
function fun1(no){
    return new Promise(function(res,rej){
        if(isNaN(no)){
        rej("Error");
        }
        else{
        no++;
        res(no);
        }
    })

}
function fun2(){
}
fun1('abc').then(function(dt){
    alert(dt);
     fun1(dt).then(function(dtt){
```

```
        alert(dtt);
        fun1(dtt).then(function(dttt){
        alert(dttt);
        },function(err){
        alert(err)
        })
        },function(er){
        alert(er);
        })
    },function(e){
    alert(e)
    })
</script>

---------------------------------

sal - 1L - 4L
    Tds  - no
    4L- 8L
    Tds - 10%
    8L - 12L
    Tds - 20%
    12L - 16L
    Tds - 25%
    > 16L
    Tds - 30%


Hra - Tds - 5000-7000
    Hra - 5000
    Tds >7000 <10000
    Hra - 8000
    Tds > 10000 < 12000
    Hra - 10000
    >12000
    Hra - 15000

Pf : Hra - 7000 && 8000

    pf-1000
    Hra>8000 < 9000
    pf-1500
    Hra >9000 <10000
    pf-2000
    Hra > 10000
    pf-30000


----------------------------

Promise.all() : This function executes multiple promise objects and returns success value if all are executed successfully. If any one fails it returns failure value and call error callback.

<script>
var p1=new Promise(function(rs,rej){
    rs("From p1");
})
var p2=new Promise(function(res,rej){
    res("From p2");
})
Promise.all([p1,p2]).then(function(dt){
console.log(dt);
}).catch(e=>console.log(e))
</script>

--------------------------

Promise.race() : Executes the first occurred promise if it is resolved / rejected one.
```

```
<script>
 var p1=new Promise(function(res,rej){setTimeout(()=>rej("From p1"),100)})
 var p2=new Promise(function(res,rej){res("From p2")})

 Promise.race([p1,p2]).then(function(dt){
    console.log(dt);
 }).catch(e=>console.log(e))
</script>

----------------------
Promise.any() : Returns the first resolved promise object.

<script>
 var p1=new Promise(function(res,rej){res("From p1")})
 var p2=new Promise(function(res,rej){res("From p2")})

 Promise.any([p1,p2]).then(function(dt){
    console.log(dt);
 }).catch(e=>console.log(e))
</script>

-------------------------
Promise.allSettled() : To get the information of all api services (resolve/reject)
<script>
 var p1=new Promise(function(res,rej){rej("From p1")})
 var p2=new Promise(function(res,rej){res("From p2")})

 Promise.allSettled([p1,p2]).then(function(dt){
    console.log(dt);
 })</script>

-------------------------
call() : using call we can call a function of an object by passing another object as own
object of that function.

<script>
var obj={
    uname:"scott",
    city:"hyd",
    fun1:function(x,y){
    alert(x)
    alert(y)
    var uname="John"
    alert("From fun")
    alert(uname);
    alert(this.uname);
    alert(this.city);
    }
}
//alert(obj.uname);
var nobj={uname:"Amith",city:"chennai"}
obj.fun1.call(nobj,100,200);
</script>

------------------------------
apply() : It is same as call but we need to pass array as argument

<script>
var obj={
    uname:"scott",
    city:"hyd",
    fun1:function(x,y){
    alert(x)
    alert(y)
    var uname="John"
    alert("From fun")
    alert(uname);
    alert(this.uname);
```

```
    alert(this.city);
    }
}
//alert(obj.uname);
var nobj={uname:"Amith",city:"chennai"}
obj.fun1.apply(nobj,[100,200]);
</script>

---------------------

bind() : It is same as call but returns function instead of execute.

<script>
var obj={
    uname:"scott",
    city:"hyd",
    fun1:function(x,y){
    alert(x)
    alert(y)
    var uname="John"
    alert("From fun")
    alert(uname);
    alert(this.uname);
    alert(this.city);
    }
}
//alert(obj.uname);
var nobj={uname:"Amith",city:"chennai"}
var rv=obj.fun1.bind(nobj,100,200);
//alert(rv);
rv();
</script>

----------------------

Object Oriented Concepts : JavaScript supports OOPs concepts like inheritance, constructor,....
OOPs we can use to implement applications in a structured way.

According to oops executable statements we should place inside the class.

class is collection of members. class members are properties and methods. Variable declaration inside the class we can call as
property. function declaration inside the class we can call as method.


object : Object is instance of class. class members we can access using object. we can use 'new' keyword to create class object.


constructor() : constructor is a type of method contains class name as method name. By default every class contains a pre defined constructor to create class object. we can also create user defined constructor to execute some statements at the
time of creating class object. using constructor keyword we can create user defined constructor.

<script>
    class cls1{
         sno=100;
        fun1(){
            alert("From class method");
         }
    }
    var obj=new cls1();
    alert(obj.sno);
    obj.fun1();
</script>

---------------------------------------------
```

```
<script>
    class cls1{
         sno=100;
        fun1(){
            alert("From class method");
         }
         constructor(){
            console.log("Cons exec....")
         }
    }
    var obj=new cls1();
    //alert(obj.sno);
    //obj.fun1();
</script>
```

---

this : using this keyword we can access the members of class from the methods of same class.

```
<script>
    class cls1{
        uname="scott";
        fun1(){
            var uname="Alex"
            alert(uname);
            alert(this.uname);
        }
    }
    var obj=new cls1();
    obj.fun1();
</script>
```

Inheritance : It is a concept of inheriting the members of a class from another class. using extends keyword we can inherit the members of a class from another class.

We have different types of inheritances

1) Single Inheritance
2) Multiple Inheritance
3) Multi level Inheritance
4) Hybrid Inheritance
5) Hierarchical Inheritance.

Single Inheritance: In this concept we have only 2 classes one is base class another one is derived class.

Ex:

```
<script>
class cls1{
fun1(){
alert("From class1");
}
}
class cls2 extends cls1{
fun2(){
alert("From class2");
}
}
var obj=new cls2()
obj.fun1();
</script>
```

Multiple Inheritance : In this concept a class can have multiple parent classes. JavaScript does not support this concept.

Multi Level Inheritance : In this concept a class can behave as both base class and derived class.

```
<script>
class cls1{}
class cls2 extends cls1{}
class cls3 extends cls2{}
</script>
```

Hierarchical Inheritance : In this concept a class can have multiple derived classes.

```
<script>
class cls1{}
class cls2 extends cls1{}
class cls3 extends cls1{}
</script>
```

Hybrid Inheritance : It is collection of single, multiple, multi level and hierarchical inheritances. JavaScript does nit support this concept.

---

super : using this keyword we can access the members of parent class from the derived class.

```
<script>
class cls1{
fun1(){
alert("Fun1 from class1");
}
}

class cls2 extends cls1{
fun1(){
alert("Fun1 from class2");
}
fun2(){
alert("Fun2 from class2");
this.fun1();
super.fun1();
}
}
var obj=new cls2();
obj.fun2();
</script>
```

Ex:

```
<script>
class cls1{
sno=1234;
    fun1(){
    alert("fun1 from class1")
    alert(this.sno);
        let funinner=()=>{
        alert("From inner function");
        alert(this.sno);
        }
    funinner();
    }
}
var obj=new cls1();
obj.fun1();
</script>
```

Async / Await : It is a concept of working with asynchronous functionalities. Async we can apply to a function to work with asynchronous functionalities and await we can use to wait until the current asynchronous functionality is completed.

With out async we can not use await.

Ex:

```
<script>
    async function getData(){
        var final_data=await fetch("https://restcountries.com/v3.1/all");
        console.log(final_data);
    }
    function fun2(){

    }
    getData();
    fun2()
</script>
```

---

event : event is an object to get the current event information. It provides many properties related to the current event which if fired.

Events are 2 types

1) Mouse Events : These events fires when we perform any action with mouse
    Ex: onclick,onmousedown,onmouseover,....

2) Keyboard Events : These events fires through selected character of keyboard.
    Ex: onkeyup,onkeydown...

Properties of mouse events.

clientX : To get the x position of mouse.
clientY : To get the y position of mouse.

These two properties are available in onclick,onmousemove,...

Ex:

```
<script>
    function fun1(e){
        var xpos=(e.clientX);
        var ypos=(e.clientY);
        document.getElementById("div1").innerText=`X pos is ${xpos} and Y pos is ${ypos}`;
    }
</script>
<body onmousemove="fun1(event)">
    <div id="div1">

    </div>
</body>
<style>
    body{
        margin:0px;
    }
</style>
```

---

button : using this property we can get the information of mouse button clicked by user. Ig user click on left button if holds value 0 , if center button value is 1, if right button value is 2. It is available in onmousedown event.

Ex:

```
<script>
    function fun1(e){
        alert(e.button)
    }
</script>
<body onmousedown="fun1(event)">

</body>
```

Ex: 2

```
<body>
    <img src="all.jpg" width="200px" onmousedown="fun1(event)"/>
</body>
<script>
    function fun1(e){
        if(e.button==2){
            alert("Can not copy")
        }
    }
</script>
```

---

type : to get the type of event which is executed.

```
<body>
    <img src="all.jpg" width="200px" onclick="fun1(event)"/>
</body>
<script>
    function fun1(e){
        alert(e.type);
    }
</script>
```

---

key : To get the information of key clicked by user. It is available in keyboard events like onkeydown/onkeyup

Ex:

```
<body>
    <input type="text" onkeyup="fun1(event)"/>
</body>
<script>
    function fun1(e){
        alert(e.key);
    }
</script>
```

---

onkeydown : This event triggers when we press the character in keyboard. First it calls function later it displays key in the textbox

onkeyup : This event triggers when we release the key. first it prints character in textbox later it calls the function.

Ex:

```
<body>
    <input type="text" onkeydown="fun1()"/>
</body>
<script>
    function fun1(){
        alert("Func called")
```

```
    }
</script>

--------------------------
keyCode : To get the keycode value of input character. keyCode is same as ASCII but it is common for both alphabets.

ASCII
------
A - 65
B - 66
Z - 90

a-97
b-98
z-122

-----
0-48
1-49
9-57

-----

space - 32
enter - 13
bksp - 8
tab - 9

Ex:

Note : keyCode returns purely ASCII values when we use onkeypress event.
------------------------------------------------------------

Ex:

<script>
 function fun1(e){
    document.getElementById("div1").innerHTML='';
    if(e.keyCode < 48 || e.keyCode>57){
        document.getElementById("div1").innerText=("Enter Number")
        e.preventDefault();
    }
 }
</script>
<body>
    <input type="text" onkeypress="fun1(event)" />
    <div id="div1"></div>
 </body>

------------------------------------------------------------

Event Bubbling/Trickling : It is a concept of execution of  parent element events when we perform an action on child element.

Bubbling is the concept of calling events from down to up direction (bottom to top).

Trickling is the concept of calling events from up to down(top to bottom);


By default events flow if bubbling.

These two flow execute every time but events execute in the specified flow.

The first executable flow is trickling later bubbling.

if third argument is true that event will call in trickling flow, if false it will call in bubbling flow
Ex:
```

```
<body>
    <div id="div1" >
        This is div1
        <div id="div2" >
            This is div2
            <div id="div3">
                This is div3
            </div>
        </div>
    </div>
</body>
<style>
    div{
        padding:10px;
        border:1px solid silver;
    }
</style>
<script>
    document.getElementById("div1").addEventListener('click',function(){
        alert("div1 called")
    },true)
    document.getElementById("div2").addEventListener('click',function(){
        alert("div2 called")
    },false)
    document.getElementById("div3").addEventListener("click",function(){
        alert("div3 called")
    },true)
</script>

-----------------------------------------------
event.stopPropagation() : using this function we can stop the flow of bubbling and
trickling

Ex:

<body>
    <div id="div1" >
        This is div1
        <div id="div2" >
            This is div2
            <div id="div3">
                This is div3
            </div>
        </div>
    </div>
</body>
<style>
    div{
        padding:10px;
        border:1px solid silver;
        background-color: white;;
    }
</style>
<script>
    document.getElementById("div1").addEventListener('click',function(e){
        alert("div1 called")
        this.style.backgroundColor="lightblue"
        //e.stopPropagation();
    },false)
    document.getElementById("div2").addEventListener('click',function(e){
        alert("div2 called")
        this.style.backgroundColor="lightgreen"
        e.stopPropagation();
    },true)
    document.getElementById("div3").addEventListener("click",function(e){
        alert("div3 called")
        this.style.backgroundColor="yellow"
        //e.stopPropagation();
```

```
    },false)
</script>

--------------------------------------------------------------------------------
rest : using rest operator we can get rest of the array elements into variables.
<script>
    var arr=[10,20,30,40,50];
    var [x,y,...z]=arr;
    alert(x);
    alert(y);
    alert(z[0]);
</script>

spread : To spread the values of array into multiple variables

<script>
    var arr=[10,20,30,40,50];
    function fun1(x,y,z,a,b){
        alert(x);
        alert(y);
        alert(z);
        alert(a);
        alert(b);
    }
    fun1(...arr);
</script>

------------------------------------------------------------------------
default : To set default value to the argument of function.

<script>
    function fun1(x,y=333){
        alert(x+y);
    }
    fun1(100,200)
    fun1(1)
</script>

---------------------------------------
protocols:  protocol is set of rules to transfer the data from one location to another location.

protocols are 2 types
1) state full protocols
2) state less protocols

state full protocols : These protocols can maintain the state of application. Means they can transfer the data of one page to another page. These protocols we can use in windows applications.

ex: tcp/ftp,...

state less protocols : These protocols can not maintain the state of application. Means these protocols can not transfer the data of one page into another page. We can use these protocols to work with we applications. Because these protocols dont carry the data of one page to another page so that the transmission speed is more.

ex: tcp/ip, http, https,...


state management : using this concept we can maintain the state of application. This concept provides objects like
cookies
sessions
local storage
session storage
....

JavaScript supports localStorage object with many methods to maintain the state of
```

application.

getItem() : using this method we can read the data of local storage

setItem() : To create a property in local storage

removeItem() : To remove a propery of local storage

clear() : To clear all the properties of local storage

Ex:
---
p1.html

```
<script>
     localStorage.setItem("x","100");
     localStorage.setItem("y","200")
    alert(localStorage.getItem("x"));

</script>
<body>
    <h2>This is p1</h2>
    <a href="p2.html">Open P2</a>
</body>
```

p2.html

```
<script>
alert("from p2");
alert(localStorage.getItem("x"));
//localStorage.removeItem("y")
localStorage.clear();
</script>
<body>
  <h2>This is p2</h2>
</body>
```

---

session storage : It is same as local storage but the data of session storage will destroy once we closes the browser.

Note : Local storage data is available in browser even we close, but not session storage
       local storage we can access from another tab but not session storage

---

closure : closure is an object to maintain the data of outer function which we can access from inner function.

The variable of outer function will destroy when the execution of that function is completed.
but if we access the outer function variables from inner function(lexical scope) then the data should prevent in an object, that object is closure.

console.dir() is a function to display the closure data