JS Engine
    is the execution unit of browser
    and it is resposible for the code execution only at the browser side
    each and every browser have differnt JS Engine

    example :

Browser	Js Engine
chrome	V8
FireFox	SpiderMonkey
Safari	JS core
MS Edge	Chakra

    parser = check for the syntax mistake it will return abstract syntax tree is AST
    AST = it is just the result of the parser
    interprter : it will check the code line by line
    JIT = just intime compiler
    Byte code : it will understand by the Machine only
    then code execution will done

Window :
     it is the supermost object in javascript
     it refers to the browser window
     it will have differnt method and properties
     if we declare the variable using the var then it will also get store in the Window object

    (IMP) if we want to acess any properties then we can access with window as prefix or we can use directlly also only for the window object
     example = window.alert();

Note : all the methods are the childs of the window object (prompt() , document() , etc)

//.....................................//

this Keyword:
    if you declare it globally then it will refer to the Window Object
    so we can access the global scope varaible in the local scope using
    Syntax :
            this.globalVariableName

//...//

GEC : (Global Execution Context we have two phase)
when we create the javascript code then GEC will get created
phase 1 = varaible declaration or function declaration phase

phase 2 = It will initialize the value to the variables and also checks the function call

then for the each function calling statement it will create the Function Execution Context and it will also have the 2 phase as similar to the GEC

call stack : it will give the info about the which function running now or to track the which functions are getting executed

//.....//

Nested Function : function inside the function

//.....//

Closure(very very IMP Question)
it is the scope or the memory allocation which gets created when you access the outer function variable inside the inner function

or whenever we will try to access the outer function variables inside the inner function then the closure will get created for the outer function

//.....//

Lexical Scoping :

variable hosting
  moving variable declaration to the top is known as varaible hosting
  before the code get execute it will first get load
  means the taking the varaible declaration to the top

Note : it will return undefined for the only var variable but for let and const it will throw the error (Uncath referance error)
        because of the (Temporal Dead Zone)   very IMP
        1)means for script scope the value will be unavailable so you will get error here
        2)and for the global scope and local scope varables will be always undefined

//.....//

function hosting
      means the taking the function declaration to the top
      it is only applicabled for the only named function
      it is not compulsory to call the function after declaration you can call before declaration

//.....//

Array :
     notes in phone in image format
     array length is not fixed in javascript
     after deleting the element the length will remain same but index position will get deleted
     if we create the any array then parent will be the array object
     it is like inheritance we can access the properties and methods of an parent

differant way :
     1)literal way
     2)using array constructor (using the new and arrayConstructor(length of the Array))
              IMP = here it will for 1 value it will treat the length of the Array and more than one value it will treat as the array elements
                  1)new arrayConstructor(length of the Array); or we can directlly pass the values
                  2)new arrayConstructor(Array elements directlly);
                          example: var arr = new arrayConstructor(10,20,40,54,78,69,85);

 3)using Array.of(pass the array elements);

1)of is the one method
2)directlly pass the array elements in Array.of();
3)it will treat only as the element of the Array

example:

```
var x = Array.of( "Hello" , 20 , 1n , [10 , 50] );
```

Note : most time we will use the literal way to create the Array element

//.....All methods are the Non static meyhods call with the object referance .....//

Push() method In Array :
it will return the length of the modified array
it will add the elements at the last
it will help to add the multiple elements

unshift() method
it will add the array element at the beginning
you can add the one or more elements like push method
it will also return the length of the Array

pop() method :
it will remove the end of the Array element
we can only able remove the one element at the time
if the array is empty then it will give the error
it will return the deleted array element

shift() method :
it will only remove the only one element
it will remove the array element from the beginning
and it will return the beginning deleted element

slice() method:
it will extract the part of the array and return the new extracted elements in the form of the Array
it will take the two arguments
it will not modify the orignal Array
it will return array hence we have to store that in the one variable then we can print that part of the Array

```
        if we are passing only one argument then fom that index it will return the
all array elements in the form of the Array
        we can also pass the negative values as an argument then it will consider
from the last index from the orignal Array .

        example:
              ar.slice(start-index , last-index - 1);   //it will not consider the
last index value


splice() method :
            it will take three or more arguments
            it will affect the orignal array it will make t he changes in  the same
array
            when we want to add the elements along with that delete the element
then we can go with the splice method
            it will add the array elements at the position for which we have
already want to delete the array element
            first two arguments are mandatory to mension

   Arguments :  arr.splice( start-index , delete-count-element ,
element-that-we-want-to-add);




//..............for  loops  ............................//

Note : in javascript we can able to itearate on the two data structure that is :
                    String and Array

for loop : it is same as the java


find() method:
          it is the higher order function
          it will take one function as an argument
          the function that we are passing that will take the three parameters
          (array-value , array-index , orignal-array )   //that function will take
this three parameters
          it is the array method
          it will return the first statisfyed element
          it will itearate over the array and return the first element  for the
satisfied condition
          it is the non static method so call it with the help of the object
referance

        Syntax :
                ArrayName.find((array-value , array-index , orignal-array ) => {
```

```
                    console.log(array-value);
                    return (condition ) //like array-value > 50;
                })

Note : if no array element satisfy the condition it will return the undefined

//.........................//


findIndex() method:
            it is same as the find method but it will return the array element
index
            its return type is number

        Syntax :
                ArrayName.findIndex((array-value , array-index , orignal-array ) =>
{
                    console.log(array-value);
                    return (condition ) //like array-value > 50;  //it will return
the index of that element
                })

Note : if no array element satisfy the condition it will return the -1


//.........................//

filter() method:
            it is also the higher order function
            it will always return the element of the array
            it will go and check for each and every Array element
            it will return one Array and in that array it will return all the
satisfied element
            return type of the filter method is array

        Syntax :
                                    //used anaynomous function as the parameter
                ArrayName.filter(function(array-value , array-index , orignal-array
) {
                    console.log(array-value);
                    return (condition ) //like array-value > 50;  //it will return
the hole array elements for which the codition is true
                })



Note : if no array element satisfy the condition it will return the empty array
Note1: Very IMP it is not mandatory that we have to pass the all three parameters
in the function
      filter and map method will not affect the orignal array it will return the
```

new array

map() method:
          it is also the higher order function
          if we want to perform the same operation for each and every array element then we use map() method
          it will return one array
          it will perform the opearation and it will add the array elements in new array and return that array

        Syntax :
                                    //used arrow function as the parameter
                ArrayName.map((array-value , array-index , orignal-array) => {
                    console.log(array-value);
                    return (condition) //like array-value + 50; //it will return the hole array elements by adding 50 in it for each and every array elements
                })

Note : if we want to perform the opearation for each and evey array element like addition , subtraction , division , multiplication it will do the opearation and add the new value in the new array and return that new array

some() or every()
          both are the higher order function
          here also we have to pass the condition
          it will return the boolean
          both are the non static

          some():
                  it will return true if the any one element satisfied the codition else it will return false
                  once the condition is satisfy it will not check for thr further elements

          every():
                   it will return true if the each and every element satisfied the codition else it will return false

I have to go through
reduce() method:
        if you want to add the array elements then we will use this

```
        it is also the higher order method
        it is also the non static method
        it will take one call back function as an argument
        it will reduce the array and return the single value
        it will itearate from left to right
        default value of the accumalater will be 0
        if we want to pass the initial value for the accumalater then at the
second-argument pass it then it will go inside the Accumalater-value


        Syntax :
                                //used arrow function as the parameter
                ArrayName.reduce((Accumalater-value , Current-value) => {

                        return Accumalater-value + Current-value;
                } , second-argument)




reduceRight()  method:
        it will itearate from right to left
        all other is same to the reduce() method only



forEach() method:
|           it is also the higher order function
            it is the inbuilt method in javascript
            and it is used to iterate over the array
            it is used to iterate over the array


        Syntax :
                                //used arrow function as the parameter
                ArrayName.forEach((array-value , array-index , orignal-array ) => {
                    console.log(array-value);

                });

Note Imp : diff between map() and forEach();

map() = it will return the array
forEach() = it will return the void (hence it will return undefined)



//.........................................///

5 ways to itearate over the array
```

```
            1) normal for loop
            2) for in loop
            3) for of loop
            4) map();
            5) forEach();

2) for in loop :
            used to iterate the index of the array
            (identifier , in , ArrayName);


    Syntax:
            for(identifier , in , ArrayName){
                //here we can access the only index of the element
                console.log(index);
            }




3) for of loop
            used to itearate over the values present inside the array

    Syntax:
            for(identifier , of , ArrayName){
                //here we can access the only array element
                console.log(value);
            }

//.............................//


indexOf() method:;
        it will return t he particular index position of the particular array
element
        it will only take the one parameter  that is array element that we want to
find index of that element
        it will return -1 if array element is not present


    Syntax:
            ArrayName.indexOf((mandatory argument) array-element ,
(optional-argument) that specify which index I should Start for searching that
element);


lastIndexOf() method:;
            it will take the only one argument
            that is the search the value array element
```

```
            and it will search till the last indexOf the array
            it will return -1 if array element is not present

    Syntax:
            ArrayName.lastIndexOf( array-element );



fill() method:;
            if we want to fill the same value for all the array element then we
will use fill()
            it will take three arguments (value , start-index , last-index ) ; //it
will not consider the last index
        it is not mandatory to pass the values for the all array elements

    Syntax:
            ArrayName.fill(value , start-index , last-index );



include() method:
        it will return the boolean value
        if the value is present in the array it will return the true or it will
return the false



Conat():
        it is used to merge two or more arrays
        it will not affect the orignal array



    Syntax:
            let a = ArrayName.concat(arrays names how much we want to concatinate
);







///.....................remaining ......................//



sort()

flat()
```