```
    variable => datatype => datatype conversion => operator

        operation

        operand:
            the value which we are performing operation on it.

        operator :
            - An operator is a special symbole usded to perform operation on operands (value
and variable)


            i) Unary Operator
                => The operators which perform operation on only one value/ operand.
                    a) increament opertor   3++,
                        ++
                            i) pre-increment

                                ++operand

                            ii) post-increament

                                operand


                    b) decreament operator  3--,
                        --
                            i) pre-decreament
                            ii) post-decreament
    ========================================================================================
==============

            ii) Binary Operator
                => The opertors special symbole which perform operation on two values. +,-,*

                    a) Arithmatic Operator
                        => The operator which helps to perform mathematical operations it
called arithmatic operator.
                        => there will be two operands and it will arithmatic operation it
called arithmatic operator.
                        +, -, *, /, % ,**

                    b) Assignment Operator: (=)
                        => It used to assign a value to variable.
                        => it used to assgin a value to a variable of anather.

                    c) Comparision/ Relational Operator :
                        => It used to determine equality or difference between value or
variable.
```

```
                        ==  => it compare the value only
                        ===  => it compare value and datatype also.
                        !=   => if the value is not equal then output will true.
                        !==   => if the value is not equal also the datatype is not equal.
                        <    => if the value is less than other value.
                        >    => if the value is greater than other values.
                        <=   => if thhe value should less than or equal to other value
                        >=   => if the value should greater than or equal to other value.


                    d) logical operator:
                        => Logical opeerators are used to dtermine the logic between the
variable or value.

                        &&   => it check both/ all the condition/logic true or not
                             => if any one logic is not true it will returns : false
                             => if both / all the condition/ logics are correct it retuns :
true

                        ||   => if there is any one condition/ logic is true it will returns:
true
                             => if both / all the logics/conditions are wrong/false returns:
false

                        !    => it will returns false if the logic /condition is correct
                             => it will returns true if the logic / condition is not correct

=============================================================================================
======================

        iii) Ternary Oprator
            => The opertors which perform operation on three values.
            => The ternary operator used to simplified conditional operator ex. if, if else

            mark>35 = pass
            mark<35 =fail

            syntax:
                consition ? expression for true : expression for false;
                mark>35 ? "pass" : "fail"
            => ? : to check the condition if true or false.
            => if the condition is true => left side of colon expression will print.
            => if the condition is false => right side of colon expression will print.

            * to work with multiple condition.

            syntax:
                condition?"expression of true" : condition2 ? "expression for true for
condition 2" : condition 3 ? "true condition3" :condition 4 ? "true4" : "false"
```

```
    =========================================================================================
============================
    variable => datatype => operater => statements
    =========================================================================================
============================

    Statement:
        => It is combination of variable, data type, operator.
        => Javascript statments are used to give the instruction to browser for the action.
        => statements are saperated by semicolon (;)

        i) Declearation statement
            => where we declear variable, function, array, object that statement/line of code
called declearation statement.

        ii) Arithmatic statement
            => where we do arithmatic operations/ calculation that statement called as
arithmatic statement.


        iii) conditional statement :
            => it works on the condition / logic and control the flow of coding
            => it decide and instruct to the browser if the condition / logic is correct how
to work and if not correct then how/ what to action.

            i) Branching statement
                => it is very important to exicute program with respect to the cetain
condition.
                => Using branching we can control program flow.
                    a) if statement
                        syntax:

                            if(condition/logic){
                                return statement.
                            }

                                => if the condition is true it will exicute return value.
                                => if the condition is false it will stop there.

                    b) if else statement
                                syntax:
                                if(codition){
                                    return statement
                                } else {
                                    return statement
                                => if the condition/logic is true it exicute if defination
block
                                => if the condition / logic is false it exicute else
defination block

                    c) else if lader :
                                syntax:
                                    if(condtion){
                                        return value
                                    }
                                    else if(condition2){
                                        return

                                    }
                                    else if(conditio3){
                                        return value
                                    } else
                                        { false value}

                                => it help us to get more than one conditions in same
program/ statemnt.
                                => where it get true value it exicute defination block
                                => if it get false value it exicute else defination block



                                if(){}              if true print if block
                                if(){} else{}       if true print if block or if false print
else block
                                if(){} else if(){} else if(){} else{} if true print if block
if condition false check else if 1, if false check else if 2, if all false print else


                switch case:
                        => use the switch statement to select one of many block of code to be
exicute.

                        => it find exact match / input.



                        english1  hindi2   marathi3


                        syntax:
                            language= ;

                            switch(language){

                                case 1:
```

```
                                language="English"
                                break;

                                case 2:
                                language="Hindi"
                                break;

                                case 3:
                                language="Marathi"
                                break;


                                default:
                                languuae="wrong choice"


                            }


                            we have ,

                            case => which match to exicute.
                            break => if got exact match break the process and exicute match
block.



                        1 => sms pack
                        2 => internet add on pack
                        3 => talktime
                          => please select one of above only

                        wel


                        1,2,3,4,5,7
                            console()


            ii) Interation/ looping /repeatation:
                    Loop :
                        => Loop can execute a block of code a number of times.
                        => Loops are easy to use when we want to run the same
                            code over and over again, each time with a diffrent value.
                        => generally loop use for working with arrays.

                    for(expression 1/ start,
                        expression 2/ stop,
                        Expression 3/ increament | decreament){
                              (code to repeatation)
                    }



                    * pRINT welcome 10 times using for loop...
                                1
                        initialization    condition     increament/decreament
                            start,           end,          increament
                    for(  let welcome=1;   welcome<=10     ++        /  -- ){

                            console.log(welcome);
                        }

                    =======================================================

                    while loop :
                        => The while loop, loops through a block of code as long as a
specified condition is true.

                        syntax :
                            for(expression1,exp2,exp3){

                                console.log("welcome");

                            }

                        =>

                            exp1/(declearation)
                        while(exp 2/condition{

                            console.log("welcome");

                            exp 3 (increament/decreament)
                        }


                        var i=1;
                        while(i<=10){

                            console.log("welcome");
                            i++
                        }

                =======================================================
                do while
                    => The do while loop is the varient of while loop.
                    => this loop will exicute the code atlist once, before chacking if the
condition is true
```

```
                    syntax:
                    declearation exp 1
                    do{
                            console.log("welcome");
                            i++ exp3


                    }

                    while(condition exp2);


                    for(exp1,exp2,exp3){

                        code

                    }


                    exp1
                    while(exp2){
                        code
                        exp3
                    }



                    exp1
                    do{
                        code
                        exp3
                    }while(exp2)



            iii) jumping




    ========================================================================

    statement=> it is the combination of variable,datatype and operator


    function:
        => it is a combination of statement.
        => function is a block of code which designed to perform a particular task.

    Adavantages of javascript function:
        => code reusibility.
        => function exicute the code when we call it.
        => without calling function not exicute.
        => we can call the function many time to reuse the code.
        => It makes our program compact.
        => we don't need to write many lines of code each time to perform a common task.

        let num1=10;
        let num2=20;
        let num3=30;

        let total= num1+num2+num3;

        console.log(total);

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




        syntax:
            i)  function declearation=> function addition();
            ii) function defination => function wel() {code to exicute }
            iii)function calling => wel()

    simple function :
        function functionName(){
            code to exicute
        }

        functionName()





    Paramiterize function :
        => we can call function by passing arguments.
        => we should pass the arguments in the parafnthesis of function
        => we can pass vales for argument in the paranthesis of function calling.
        => function arguements are the values recieve by the function when it is invocked.
        => we parameterized function we can perform same task with different values.




    function functionName(a,b,c){

            console.log(a+b+c);

    }
    functioName(10,20,30)
```

```
    i) print your statement using simple function calling 4times
    ii) create a function with parameter for addition five number with defferent values each
time.
     ==========================================================================================
=======

     function => comination of statement.
     function feature/adv =>
     function => i) function declearation ii) function defination iii) function calling

     type of function=>
     i) simple function
        function fName()|{}|fNameI()

     ii) Paramiterize function
        function fName(a,b,c){}fName(23,"s",34)

     iii) Return function / function with return value=>
         => we can call function that return a value and use it in our program.
         function fName()
         {
            return "deposite"
         }


        function total(){
            let sub1;
            let sub2;
            let sub3;

            return sub1+sub2+sub3;
        }


        function result(){
            console.log(total());
            if(total()>200){
                    console.log("You are Pass");

                }else{
                    console.log("You are fail");
                }
        }

        ===================================================================

         advance function :
            i) function defination :
                a) by expression
                b) anonymous function
                c) Arrow function



        a) by expression :
            => when we store a fnction in avariable it's called function define by
expression.
            => we should call the function when we define function by expression by the name
of expression.
            => we can't call function expression before function defination.
            => function expression in javascript are not hoisted unlike function declearation

            syntax :
                variable/ expression = function fName(){};

        b) anonymous function:
            => The function without name.
            syntax:
                varible=function (){}


        c) Arrow function/ fat arraow function -> =>
            => The arrow function is a new feature of ES6 version.
            => it is just a syntax of short coading not a new function.
            => if we have only one line statement for arrow function we don't need to get
{}/block

            variable =()=>{}


            a,b addiotion, sub, mult, div

            using arrow function with return value
        =====================================================================================
=======

2) Function calling :

    a) function call by value
    b) function call by reference


a) call by value:
    => when we pass primitive data at function calling it called function call by value.

b) call by ereference :
    => when we pass non-primtive data at function calling it called function call by
reference
```

```
    function aman(a,b){
    console.log("welcome" a+b);
    }

    aman(10,20)



b)



    primitive       non-primitive /reference

    string          array
    number          object
    boolean
    undefined
    BigInt
    null
    symbol

    ==================================

3.  callback function
    => when we pass function as a parameter / param it call call back function.


    function aman(a){
    a()


    }
    aman(aman2)



    function aman2(){

    console.log("Hiii i am aman2");
    }


    function aman(a){
    a

    }
```

```
==========================================================

    simple fun
    paramiterized
    function a(c,d){

    }
    (a,b,c)


    return
    function aman(){
        return "a+b"

    }

    aman()


    expression
    let a=function aman(){}
    a()

    anonymous
    let a= function (){}
    a()

    arrow
    let a=()=>{}

    function call by value
    function aman(a,b){


    }
    c=10;
    d=20
    d=40;
    aman(c, d)


function call by reference

    let arr=[1,2,4,5]

    function aman(a){

    }
    aman(arr)
```

```
call back

function aman(a) {a()} aman(aman2);

function aman2(){}

=====================================================================

syncrhromous
function aman(){}aman()
console.log("end of code")

asynchronous
setTimeout(fucntion aman(){},5000)

=======================================================================

Higher Order Function

 * For higher order fuunction ytou have to know the return function.
 * when we return a function in return statement it called higher order function.


 syntax :
    function Fnamemain()
    {
    return function fnamereturn(){}
    }

    let main=fnamemain()
    main()

=======================================================================================
IIFE = Imidiately Invocked Function Expression

    => It is a function that runs as soon as it is defined.

    syntax

    (function fname()
    {
        exicution block/code
    }() )
==================================================================================

rest parameter & spread operator

    for in loop
    rest parameter
    spread operator

    for in loop
    => This loop created to print/use object / array.

    syntax :
        for(let vName in arrName){


        }

========================================================
    Rest Parameter (ES6)
    => Rest parameter is a improved/modern way to handle various
       input/argument as parameters in a function.
    => The rest parameter syntax allows us to presents an
       indefinite numer of arguments as an array.
    => it get multiple values/argument and convert it into a array.
    => it should be use at last in your parameter list.

    function fName(...a){

    }

=============================================================================

spread operator :
=> The operator is use in combination with combination with
   destructoring a array or object.
=> It spread the value of array or object like saperate value of parameter,
   as opposit of rest operator


function fName(a,b,c){

}
fName(...arrayName)

==========================================================

* create a parameterized function
* you have less parameters than argument
* use rest parameter and print / make addition of given number/ arguments

*create a prameterized function
* you have pass a array in argument
* but you have multiple parameter
* spread the values of array in your parameter and make addition of that values.

====================================================================================
=========================================================================================
=> Name :Mocha =>livescript=> javascript=> ecmascript ES6
=> Branden Eich => Netscape => 1995
=> static static
=> weakly dynamic | strict                        facebook => instagram => search => post
    var a =10     | var a: Number=10
    var b="aman"  | var b:string="aman"

=> variable : is a name of memory location/address where we store data.
            : default 24uweelj24= var a=10


=> basic var , let, const =====> hoisting

global, local/functional, block level

global : everywhere in the code.


i) where we are declearing the variable
ii) which scope we use let, var, const.

var a =10;
fucntion aman(){

}

sat , sun
var a=10;  => global/functional :

let a=10;

const a=10;


=========================================
datatype:
    i) Primitive:
        already existed in javascript
        a) number => to store number type data=>  var a= 23343

        b) string => to store text type data =>
            i) using double cote  : var a= "Aman"
            ii) using single cote : var a='aman'
            iii) using backtick : var a=`aman`

        c) undefined
            => not assign/initialized any value till now
            => we can initialization or not.
        d) symbole
            => if we want to make a value unique
            => symbole(2334);

        e) null
            => here we have decided/ cleared that there will no value in variable

        f) boolean
            => true and false

        g) bigInt
            => if you have more than 15 degit value then compiler not able to perform given
task
            => bigint is the solution above problem.
            => bigInt(2349233497233498732439)



    ii) None primitive
        => multiple and different type value
            i) Array[12,"aman",true,]
                => index number
                => console.log(array[1]);
                => arr=[] square / array

            ii) object{
                    name:"aman"
                    rollNo:233423
                    18+:true

                        }
                => console.log(object.name)
                => curly
    ========================================

Operator:
    operator => the symbole which perform any operation on operand
    operation => the method to give task to value
    operand => the values/data on which we perform operation using operator

    i) unary
        => one value / one operand
        => i) increament ,ii) decreament
        => i) increament ++ ii) decreament --
        => increament
            a) pre-increament => it will incease value then assign/next operation
            b) post-increament=> assign/operation will be first and then it will increase

        => decreaement
        a) pre-decreament =>
        b) post-decreaement=>


    ii) binary
        => it will perform operation on two values/operand;
        a) Arithmatic operation
            => we can perform mathematical operation using arithmatic operator
            => +,-,*,/,**,%,++,--
        b) comparison operator / relation operator
            =>  the give value are same or not | greater | less than
            => ==, ===, >,<,>=,<=,!=, !==,?

        c) assignment operator:
            => a=b; a=10;
            => =

        d) Logical Operator
            => logic
            => &&= if both logics are correct/true ,
               ||= if one condition is true output will be=true,
               ! = if logic is wrong output will = true ,
                   if logic is correct output will = false

    iii) ternary
        => it perform operation on three of more values
        => constion ? iftrue:(constion2):true:if false

==============================================================================
Statement :
    => Statement is the combination of variables, datatypes and operator.
    => Javascript statements are the commands to tell the browswer what to action performs.
    => Statements are saperated by using semicolon

    * there are three types of statements
        i) Declearation statement:
            => when we decleare variable, function, object,array
               that statements we called as declearation statement.


        ii) Arithmatic Statements:
            => Where we do arithmatic operation it caled arithmatic statements



        iii) Conditional Satements :

            => it gives the instruction to browser which is depend on any condition.

            i) branching statement :
                => if condition is true or false and exact match
                a) if
                    => if the given condition is true then only print any output;

                b) if..else
                    => if the given condition is true print if block otherwise print else
block

                c) else if lader
                    => we can use multiple cndition

                d) switch case
                    => ecxact match: case = "go" then it will print only the value of "go"
input

            ii) looping statement :
                    => loop helps to run the same code over and over again.
                a) for
                    => for(var i(start) ; condition (end)  ; increament/decreament) {}

                b) while
                    start var i=0;
                     while(condition) {

                     console.log()
                     i++;
                     }


                c) do while
                    var i=0;  start
                    do{
                        console.log(")exicution block
                    }

                    while(condition) end

            iii) jumping statement : break, contnue

=================================================================================
    function :
        => function is a block of code where we perform a particular task
        => it is a combination of statements
        => function exicutes only when we call it
        => Reusibility
        => less coding
        => it works faster

        syntax :
            i) declearation ii) defination / ini  iii) fnction calling / invock
            function aman(){--------- declearation
                exicution block   ------------- defination/ini
            }
            aman() -----------------function calling

            i) simple function
                syntax:

                    function fun(){



                    }
                    fun()

            ii) Parameterize function :
                => we can perform same task with different values at each time when we
calling function.
                syntax:
                    function fun(a,b,c){
                        exicution block
                    }
                    fun(10,20,30)

            iii) Return function :
                => it return any value where we call it

                syntax:
                    function fun(){

                    return 10+20;

                    }

                    function fun2(){
                    console.log(fun())
                    }
                    fun2()

========================================================================================

    advance function :
        a) function expression :
            => when we store any function in a variable

            let a=function aman()
            {
            }
            a()
        b) anonymous function:
            => function without name
            let a=function (){

            }
            a=()

        c) arrow function :
            => not use function keyword nor function name.

            syntax:
                let a=()=>{}
==========================================================================

        function calling:

        a) function call by value :



        Parameterized function

        a=function(a,b,c){



        }

        a(10,20,30) ======== if primitive type data =======call by value =====

                if we redefine values it get changes deferent value at both side

                let a=10;
                    b=a;
                    b=20;



    b) function call by reference

        arr=[10,20,40]
        arrr=arr

        arrr[2]=40

        a=(a,b,c)=>{
            }
            a(arr)========== if noneprimitive type data array / object ====== call by 
reference

            when we redefine values it get same changes at both side

==============================================================================================
=======

callback function :
    => when we pass function as parameter / param it called as callback function

    function aman(a){


    }
    aman(function aman2(){cons})


===========================================
synchhronous
fucntion aman()
{
console.log(10+20);
}

aman()

console.log("this is outside / end statement )

output :
30
this is outside/end statemnt



asynchronous
setTimeout(

fucntion aman()
{
console.log(10+20);
}5000)

console.log("this is outside / end statement )

output :

this is outside/end statemnt
30
```

```
=================================================
Higher order function :

return function

fucntion aman(){

    return  aman2(){console.log("this is aman2")}
}

let a=aman()

a()

or
aman()()
======================================================

IIFE (Imidiately Invocked Function Expression):

(function aman()
{
}())

=================================================================

While Loop:
    => the while loop, loops through a block of code as long as a specified condition is
true.

for(let i=0;   i>=10;   i++ ) {
    start       end     i/d

}


let i=0; ============start
while(i<=10){ =============end (condition)

    consol.log(i);
    i++    ========================(increament / dec)

}

============================================================================================
Do while:
=> the do while loop is the varient of while loop.
=> this loop will exicuted the code block atlist once,
   before chaking if the condition is true.
```

```
=> Then it is repeat the loop as long as the condition is true.

==========================================================================================

FUNCTION:
    - It is a combination of statements.
    - function is a block of code designed to peform a particular task.
    - the function exicute only when we will call it.
    - declearation |initializatio | calling
    declearation:
        function fName()

    defination
    fuunction fName(){

    }

    function calling
    fName()

Advantages of Function:
    => Code Reusibility.
    => Less Coding
    => It makes our program compact.

==================================================
Type of Function :
    i) Simple function :
        Syntax:
            function fName(){

                exicution code
            }
            fName();
==================================================

ii) Parameterize function// function with parameter
    => we can pass te argument in the paranthesis whhile we are calling the function
    => when we want to perform same task with diferent value each time then we
       should / can use parameterize function

    syntax:
    function fName(a,b,c) ============parameter
    {
    console.log(a+b+c)

    }
    fName(10,20,30)
======================================================================
* Create a function for the result of  student.
* if the marks are greater than 35 the result should be pass otherwise fail
* you will display the result of five student using same function code of parameterize
function.

==================================================================================
Return function / Function with return value:

=> We can call function that return a value and use it in our program.
=> where we call return function the value will return ther in program.
=> we have use return keyword to return a value/ create a return function.

================================================================================
Advance Javascript:

function fName(){

}
aman();


i) Function Define by expression  :
    => when we store a function in variable that variable called function expression.
    syntax:
        var eName= function fName(){}  ================= define by expression
        eName() ================= calling function by expression


ii) Anonymous Function():
    => the function defined without name it called anonymous function.

iii) Arrow Function ()/fat arrow function ->  => :

    => The arrow function is a new feature of ES6 version.
    => It is just a syntax of short coding not a new fuunction.
    => in this syntax we don't need to get the fucnction keyword nor function name.
    => If we have one line code in arrow function not neet to give curly brackets for
exicution block.
    => if you have multiple line of code then you should give the curly bracket.

    syntax:
        eName=()=>{
        }
        eName()

============================================================================================

define a function by expression and use the simple print "Hellow World"  ,
                                             parameterized fucnction print(10*5).
```

```
define a anonymous function with return value.

define a arrow function with return value and paramiterized
function with addition of two number


function :
    simple function
    parameterized
    return

define
    byexpression
    anonymous
    arrow

===========================================================================
Function calling


*parameterized function should be cleared
a) function call by value
b) function call by reference



 a) function call by value:
    => we can use the call by vlue method for calling the function
       when we are passingg primitive value at function calling as a argument.
        ex.
        number, string, symbole,boolean, null, undefined, bigInt

    => if we assign a variable primitive type value to another and if we redefine that other
variable
       the value will changed at before and after redefine.




    function call by reference:
    => whe we pass none primitive data type (array and object) at function calling
       as argument it called function call by reference.
    => we don't deal with value directly we deal with reference/address of value
       that's why if we assign a array / object to another obj/array and redefine the value
       it will same value at both side before redefine and after redefine.

============================================================================================
=======

Callback function:
    => When we pass function as a parameter/param it called as callback function.
    => we have invocked     the given parameter in block of function as a function.
    =>


Synchronous=>
    => It wait for each operation to complete, after that it will exicute thenext operation.

Asynchronous :
    => It will never waits for each operation to complete, rather it exicutes all operation
in the first go only
    => we can set the timing to invocked our function.
    => time will in miliseconds by default

    syntax :
        setTimeout(function aman(){
            console.log("i am aman fun");
        },5000)


Higher order function :
    => for heiggher order function you have to know the return function.
    => when we return afunction in return statement it called higher order function

=====================================================================================
callback
parameterized => while i am calling this function i am given the function as argument.

function aman(a){
a()
}
aman(aman2)

aman2(){
}


heigher order function.
return=> i am return the function not value.

function aman(){

    return function fun(){}
}

let a=aman()

a()

============================================================================================
Imidiately Invocked function Expression (IIFE) :
```

```
=> It is a function that runs as soon as it is define.

syntax:

(function aman(){

    console.log("hii iife");
}())
==============================================================================================
=
callback =>

aman(a){

}
aman(fun2)


fun2

==============================================================================================

Rest Parameter (ES6) :

=> It use at declearing time
=> Rest parameter is a way to hanle various input as parameters in a function.
=> The rest parameter syntax allows us to presents an indefinite number of arguments as
array.
=> It get multiple values and convert it into a array and pass the array to the rest
parameter.
=> The rest parameter must be a last parameter of a function.

==============================================================================================
========
For In Loop :
    => This loop created to print / use object and array.
    =>
        vName=0;
    for(var i in object/arrayName ){

        vName=vName+object/arrayName[i]
    }
        console.log(vName)
==============================================================================================
===============

Spread Operator (ES6) :
=> The spread Operator is use in combination with destructoring to a array or object
=> It spread the values of array or object like saperate fvalue of parameter as opposite of
rest parameter.
```

```
============================================================================================
======

i) if less parameter more vlaues/arguments: rest parameter
    => declearation line function fun(a,...b){}
    => collect multiple and convert into one value
    => it create a array like single value

    function fun(a,...b){


    }
    fun(10,20,30,40,50)



ii) less values/argument and more parameter: spread operator
    => calling line fun(...arr)
    => collect single argument/arr/object spread the values into multiple parameters
    => it break the array and spread / make multiple value.

    function fun(a,b,c,d,e,f){


    }
    fun(...arr)



    *create a parameterized function take only one parameter
    * get multiple values at calling
    * use rest parameter and print all the values in one parameter as array


    *create a parameterized function take multiple  parameter
    * get only one array in argument
    * now spread all values of array in multiple parameter using spread operator

==========================================================================================

varable, datatype, operator, statement, simple function.

advance =>

Advance Scope:
```

```
var let const

globaly var let const => global
insde of function => var let const

function(){
    let var const => functional

    {
        let const => block
        var => functional / default scope functional
    }

    {

    }


}


=========================================================================================

Hoistng:

    Hosting s a default behaviour of javascript,
    which movng declearation to the top of function/ program before execution.

    * if we declare varable, function in global code it will goes to the top of program.
    * if we declear variable in the function it will comes to the top of function.
    * it doesn't works on initalization it works only on declearation.
    * the initialization should be before use of variable.
    * if we use before / console before initialization it wll give undefned value.
    *

    maual code          =>      compiler read like

    d=20;               =>      var d;
    console.log(d)      =>      d=20;
    var d;              =>      console.log(d)

    var;


    Hoisting with let  and const keyword:

    let:
        => we we declear any variable after use wth let keyword, it throw the referenceError
        => can't use variable before initialzation
        => when we decleare a variable with let keyword after using it goes top due to
hoisting
           but it also associate with temperal dead zone(TDZ)
        => according to the javascript rules
        * we can't intialize a uninitualize variable



        const:
            => we can't decleare saperatly to the const so it is not possible to use before
               initializing the const variable


============================================================================================
======
Lexcal scope :
    => In javascript the inner function / The Child function get the access of a variable
whch is
       decleare and defined in it parent function this faclty called as the lexcal scope.


      syntax:
        function outer(){
                var vOutName=value;

                function inner(){

                        consol.log(vOutName);

                        } inner()

                    }outer()


            OutPut= value of vOutName;

            ex......................

            function fun1(){    //outer / parent function
                var a=20;

                    function fun2(){        //inner / child functiton of fun1

                                function fun3(){        //inner/child function of fun2
                        console.log(a)

                        }fun3()
                }fun2()
            } fun1()

====================================================================
Closures :

    * Generaly /in other static languages, when function excution get completed function
lost/wipeout
      the data/variable from memory.
    * but not in javascript
    * in javascript function do not wipe out
    * A closure is a function having access to the parent scope.
    * It preserve the data from outside.
    * A closure is an inner functiion htat has access to out functions variable.
    * due to closure we can access / use varible after completion of exucution of any
function
      out side of the function it called closures.

    fun1(){
    let a=10;

    console.log(a);

    }fun1()


    i) local scope
        => The access of variable wich delceared in current function.

    ii) outer function scope
        => When inner function is able to access outer function's variable it called outer
functioon scope.

    iii) global scope
        => when we decleare varable top of code / top of program globly and able to use in
any function
        it called global scope .

=============================================================================================
================
Currying:
    => A function that accepts multiple arguments.
    => It will transform functions into a series of function.
    => Where every little function will accept a single argument until all arguments are
completed.


    function aman(a){
        function aman1(b){
            function aman2(c){
                a+b+c
            }
        }
    }
    aman(10,20,30)

==============================================================================================
======
This Keyword:

    => This keyword used to referes to an object.
    => Which object this refers?
        => This keyword refers defferent object which is
           depends on How this keyword has used.
    => we can bind a object to a ths keyword using follows methods/syntax.


    i) default bindng:
        => If we use this alone.
        => if we don't bind manually then this keyword bind global object [objectWindow]

        console.log(this)

        function fun(){
        console.log(this)
        }
        fun()

        Output : window / global object

    ii) Implicit Binding/ Object Method Binding:
        => If use a function in any object as property of object.
        => in this function  this referece the object where n the function used.

        syntax:
            let obj1={

                name:"Aman"
                lName:"kamble"
                myfun:function(){
                    console.log(this)
                    }

                obj1.myfun()
                    output : obj1


    iii) Explcit binding:
        => The call() and apply() methods are predefined n javascript.
        => these methods can used to refere an object by our choice.
        => when we call the object use call() or apply () methods as follows

        syntax

        let obj1={

                name:"Aman"
                lName:"kamble"
                myfun:function(){
                    console.log(this)
                }

        let obj2={

                name:"karan"
                lName:"Kalamkar"

                }

        obj1.myfun.call/apply(objectName)


    iv) New binding:
        => in new bindng we use new keyword to bind object
        => it create an emty object for a function.

        syntax :
            function fName(){


            }
            let obj= new fName()

==============================================================================
class and object :

    Class :
        statement => function => class

        => ECMAScript 2015, konwns as ES6 introduce javascript classes.
        => javascript classes are templates for javascript object.



        Syntax:
            class className{
                this.variables1="something"
                variable 2=20

                fun(){
                    console.log(
                }

            }

            let c= new className()

            c.fun()

            => use class keyword to create a class
            => use curly bracket after class name
            => we can't prnt any statement without method/function.
            => we have to call the method of our class.
            => ex. c.fun()


custructor function :
    => constructor s a special function in javascript.
    => it excutes automatically whenever object created.
    => There is no need to call explictly or manualy.
    => It is use to fill the vallues for object property/ variable
    syntax:
        constructor(){

        }

    => if we are working different values/variable value each time you have to put the
       values of parameter of construction function in object only.
    => not need to create and call a method/funcction to get values from variables.
==========================================================================================
Iterator:
    => iterator => iteration => repeatation
    => iT IS A OBJECT WHICH IS RETURN BY symbole.iterator().
    => iterator has next() mothod which provides values of iterables.

    Loop :
        for loop
        while loop
        do while
        for in

    let arr=[10,20,30,40,50]


    => In other loop we don't have more controls.
    => let arr=[10,20,30,40,50] if we want to skip any value or use only two values
       other loops can give us this control.
    => But in in iterators gives us more controls than other loops.

    syntax:
```

```
    let arr=[10,20,30,40,50]

    let res=arr[symbole.iterator]();

    console.log(res.next())

=> it get the output with a object as {value:10, done:false}

=> in this object value= the value in array.
=> done = if we got all the values or not.
=> if we got all the values done will true otherwise it will be false.

* we can use all array as follows
let arr=[10,20,30,40,50]

    let res=arr[symbole.iterator]();

    console.log(res.next())
    console.log(res.next())
    console.log(res.next())
    console.log(res.next())
    console.log(res.next())

    output: {value:10, done:false}
            {value:20, done:false}
            {value:30, done:false}
            {value:40, done:false}
            {value:50, done:false}
            {value:Undefined, done:true}
* to get only values.
let res=arr[symbole.iterator]();

    console.log(res.next().value)
    console.log(res.next().value)
    console.log(res.next().value)
    console.log(res.next().value)
    console.log(res.next().value)

    output :
    10,20,30,40,50
---------------------------------------------
*To skip any value
    remove the next() from console. it will skip the value.


* to use loop n iterator

    let arr[10,20,30,40,50,60,70]

    let res= arr[symbole.interator]();
```

```
        let result= res.next()

        while(!result.done){
            console.log(result.value)
            result=res.next()
        }


        let i=0;

        while(i<5){
            console.log(i)
                i++

        }

================================================================
Iterable:

    => text string is iterable.
    => for of loops works on iterable datatype.
    => which datatype has symbole.iterator method inplicitly that is iterable.
    => bydefault the object is not iterable like an array.
    => we can make a object iterable by usingg some steps with object.
    => to make a object iterable.

    i) create a object and get a function in object using [symbole.iterator] key.
    ii) must be a object in above function with name iterator.
    iii) This iterator object must be return.
    iv) iterator object must contan a function with key name=next
    v) next function must return an object which contains return {value:"aman", done:false}

        let obj={
                name: "Aman"
                [symbole.iterator]: function(){
                        let iterator={
                            next:function(){
                                return {value:"aman", done:false}

                            }
                        }
                }
        }
```