***Function :***

A block in which code can be executed again and again.

Its execution can be happened whenever they are call out by specific defined syntax.

It saves you from repetitive coding again and again.

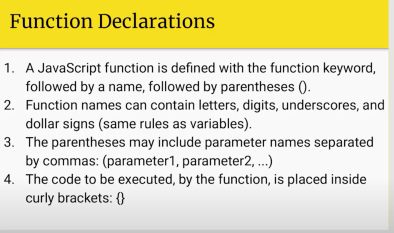
We can resuse code , define the code one time and use it multiple times we want.

Customize with different arguments and parameters.

***Function Declaration :***

Followed by a name of function with paranthesis like :

* Function funcationname ();
* Under round bracket paranthesis we pass some arguments in the function.which returns results for customization.
* Then curly brackets open close { logical block of code of function write here} ;

****

**Function example :**

**Graphical user interface, text, application

Description automatically generated**

***Function Invoke :***

in order to run function we have to call function to execute block of code in a function.

Graphical user interface, text, application, email

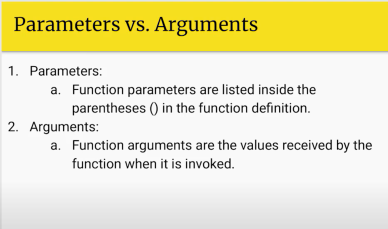
Description automatically generated

***Function parameter :***

Input given in round brackets.

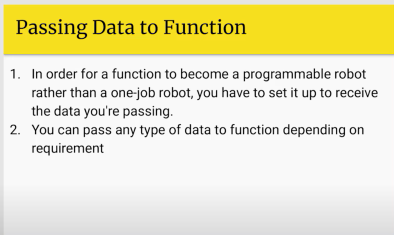
***Function Arguments :***

Passing values in function invoke.



***Passing data into function :***

Business logic same but customizable function. which takes parameters as input when parameters got changed then output changes accordingly.

 Graphical user interface, text, application

Description automatically generated

***Function return :***

Function data pass back.

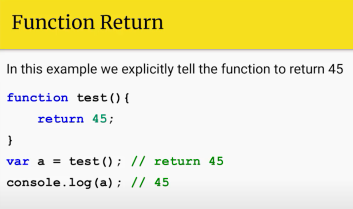
perform action afterwards sends data back to the argument.which return values to the call out wherever.

By writing keyword return

Graphical user interface, text, application, email

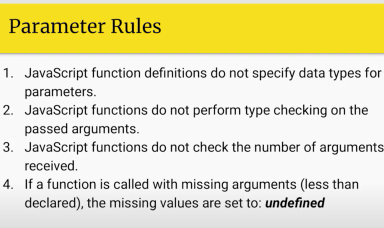
Description automatically generated

Graphical user interface, text, application, email

Description automatically generated  Graphical user interface, text, application

Description automatically generated

***Function Parameter rules :***



Graphical user interface, text, application

Description automatically generated

Examples :

// function

function sum(a,b){  // <- parameter

    return console.log(a+b);

}

function multiply(c,d){  // <- parameter

   var result=c\*d;

   console.log(result);

}

function showmessage(message){

    console.log(message);

}

//invoke function

sum(1,2); // <- arguments

sum(); // undefined

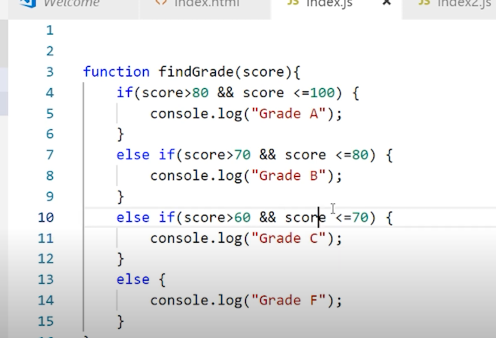
showmessage("hey there !");

multiply(3,2);

multiply(12,7);

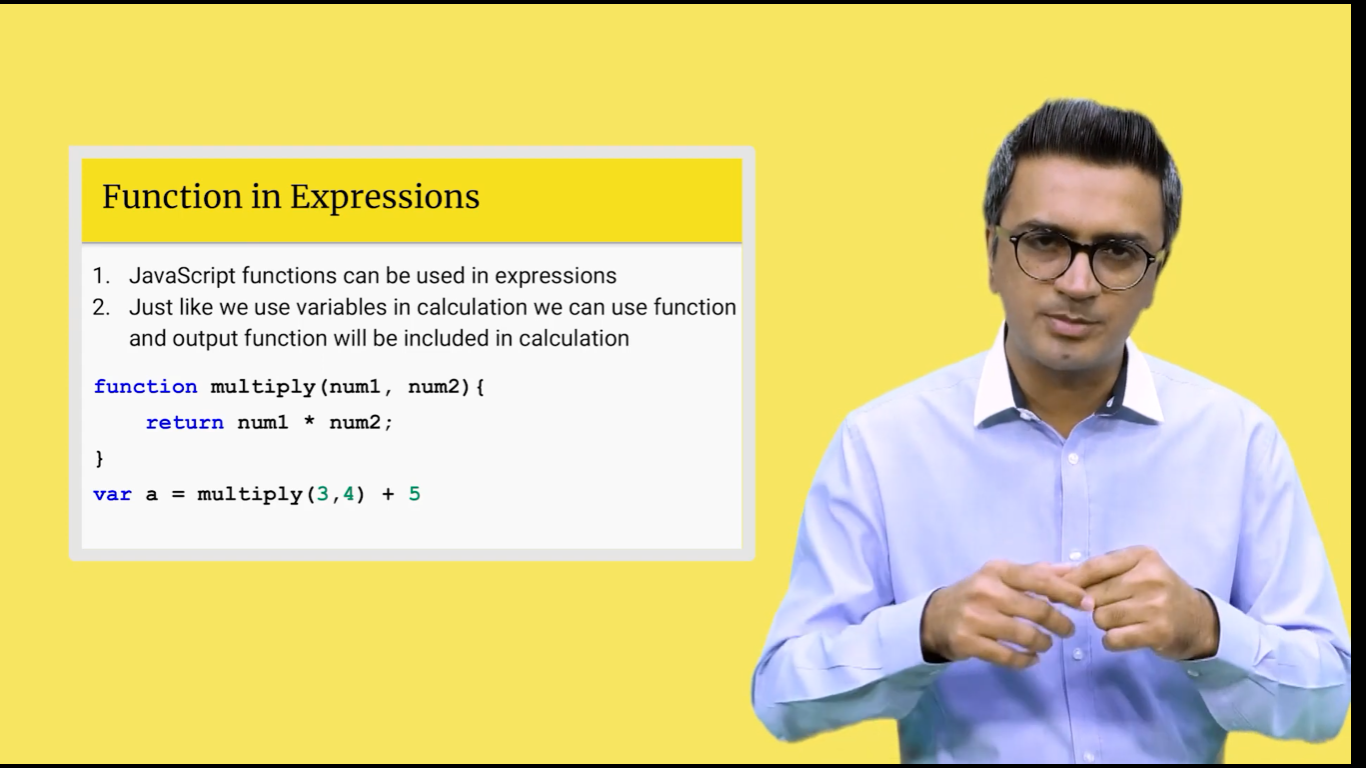
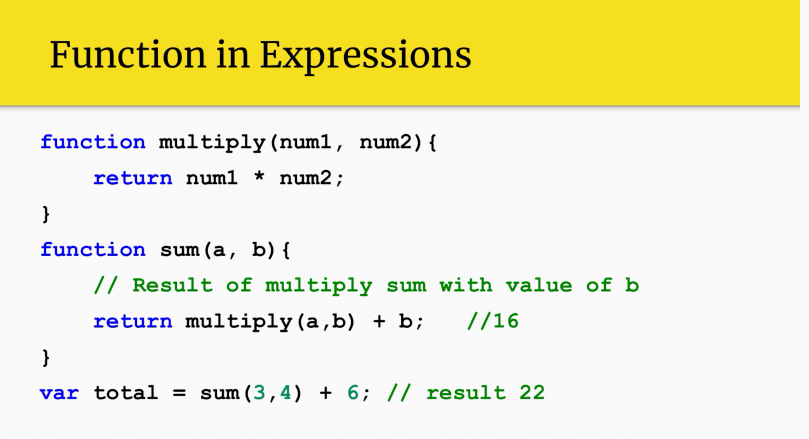
function benefit :

calling the function multiple times on same business logic.

 Graphical user interface, text, application

Description automatically generated

***Function in Expression :***

// function

function sum1(i,j){  // <- parameter

    return i+j;

}

function multiply2(g,h){  // <- parameter

 return sum(g,h)+8 // 7+8=15

}

//multiply2(9,3); //27

console.log(sum1(5,2)-3); //4 left to right calculation

example 02

// function

function sum1(i,j){  // <- parameter

    return i+j;

}

function multiply2(g,h){  // <- parameter

 return g\*h

}

//multiply2(9,3); //27

console.log(sum1(5,2)-3); //4 left to right calculation

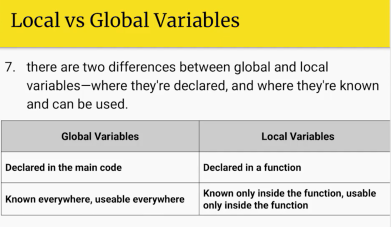
console.log(sum1(multiply2(4,5),9));  // 4\*5 = 20 , 20+9=29

***Function in Expression :***

Declare in main body of a javascript which is usually used in many funtion = global variable

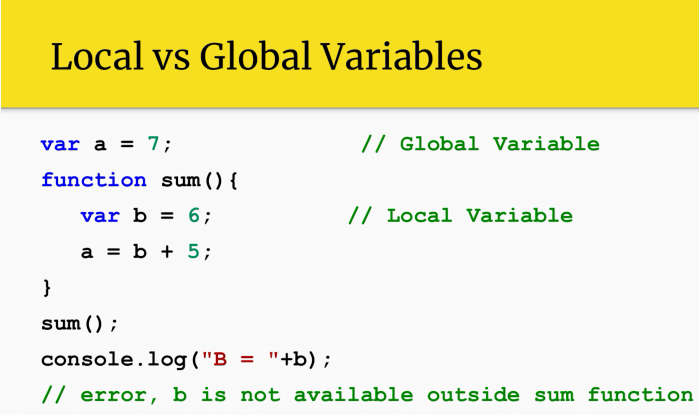
Function inside variable or parameters of a function = local variable

***Global & Local variable:***

 Graphical user interface, text, application

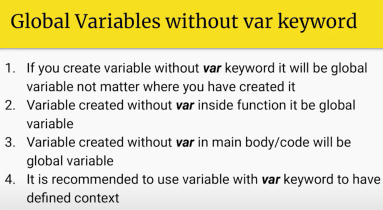
Description automatically generated Graphical user interface, text, application, email

Description automatically generated Graphical user interface

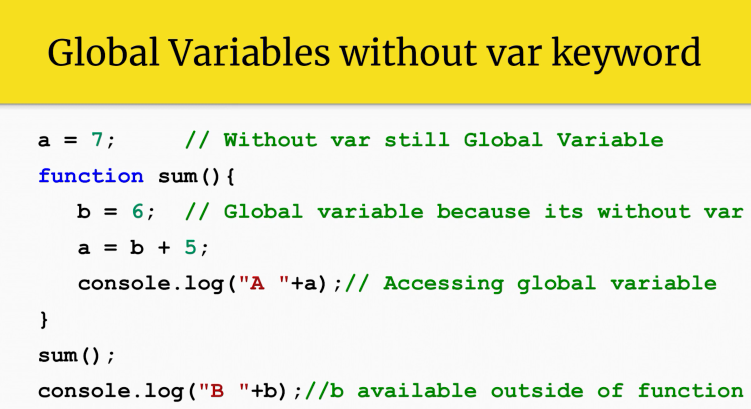
Description automatically generated with low confidence  Graphical user interface, text, application, chat or text message

Description automatically generated

***Global variable :***



Example :



Graphical user interface, text, application

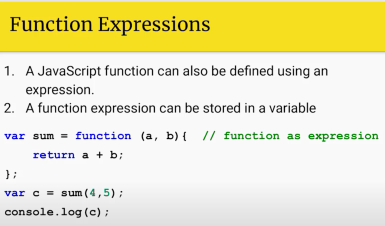
Description automatically generated

***Function Expression :***

Make a function and store it in a variable. A function can also be defined in variable.

Remember after block of function.it is necessary to have curly braces afterwards of it.

use var with variable name is not necessery



var joint = function (a,b){

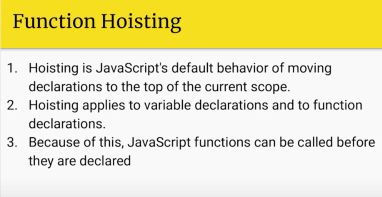
    console.log(a+b);

    return a+b;

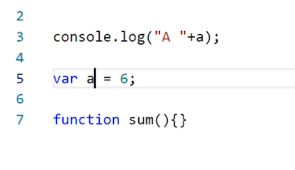
};

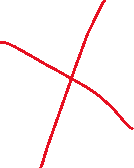
joint("jf",17);

***Function hoisting :***



Declaration goes on top. Not initialization.





Above output shows undefined.

Since a function under which declaration goes on top and works out.

Remember function is only be hoisted not expression function ultimately which cant be hoisted.

// function hoisting

var msg=energy("hoisting works ! ");

console.log(msg);  // executed.

var yo=joint(4,17);

console.log(yo);   // undefined

var joint = function (n,o){

    //console.log(a+b);

    return n+o;

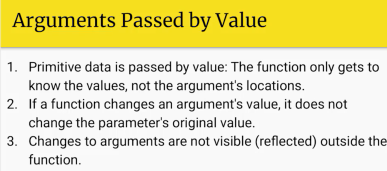
};

function energy(message){

    return message;

}

***Arguments passed by value :***



Example :

// arguments passed by value

var num=5;

var at = (a) =>  {

a=7; // a recieves 5 , it remains same no effect comes in.

};

at(num); // no change

***Arguments passed by reference :***

Graphical user interface, text, application, email

Description automatically generated

// arguments passed by reference num , string boolean -> non primitive change occur.

var arr=[1,2,4,6];

function changeout(val){

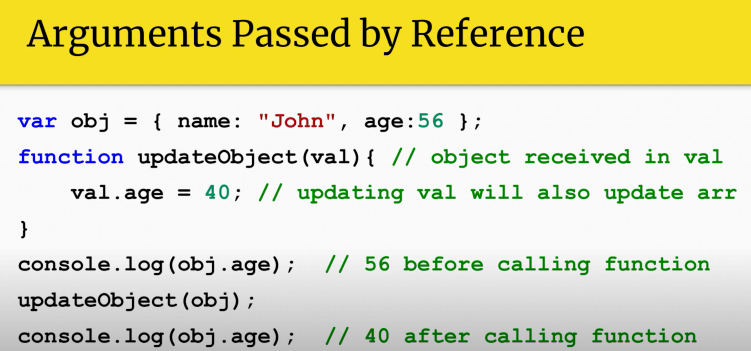
    val[2]=44; // change occur in original array despite both variables are changed but pointing the same array

}

console.log(arr[2]); // before value  4

changeout(arr);

console.log(arr[2]); // updated value 44



***Recursive function :***

// recursive function calling itself again and again until terminate condition met.

var ask=parseInt(prompt("Enter a number for factorial : "));

function recur(ask){

    if(ask<=1){

return 1;

    }

    else{

        return ask\*recur(ask-1); //e.g : 4\*(4-1)=12 -> 12\*(3-1)=24 -> 24\*(2-1)=24

    }

}

console.log(recur(ask));

***Arrow Functions :***

Arrow functions are a special way of writing functions that can be confusing at first. Their use looks like this:

Targeting function parameter:

(param1, param2) =>

{ // line 1; // any number of lines;

};

the arrow functions are useful when you want to write immediately an impelementation on the spot , or targeting a function into another function passing along with arguments.

They provide shorthand notation for writing function , they can be applied easily a arrow function underneath another function.

Arrow functions are easily stored into a variable and use it in any function.

*Lets convert simple function into arrow function*

function doingStuff(x)

{ console.log(x);

}

Conversion:

var doingstuff = x => {

console.log("arrow function variable passing with argmuemnt named x"+x);

}

doingstuff("yo");

Example 02

let add = (x,y) =>{

    console.log(x+y);

}

add(1+4);

example 03 :

let arr = ["apple","banana","Mango"];

arr.forEach(e => console.log(e));

for each loop prints elements one by one in sequence from an array.