```
//============= Function ==============

//************ Part 1 *********/

// structure of function

function name_squre(x){

    return  x * x;

}
// call the function
name_squre(5); // argument pass and call

// ****** one concern is that.
// we can use many argument in calling time
// but it work depent on parameter
name_squre(5, "Hello", 10); // but 5 only work

// ****** pass by value **********
function squre(x){
    return x * x;
}
let y = 5;
squre(y); // this pass by value
console.log(y); // 5

// ****** pass by reference ********
// if we pass object then it is PBRef inside the function
function Ref(x)
{
    // do something
    x.key = "Hasan";
}
let ref = { key: "Mehedi", }; // object
//call Ref function
    Ref(ref);
console.log(ref.key); //"Hasan" coz change for PBRef


// ================= Call Back function =========
// Call back meand big function call small function inside the BF.


function doSomething(x){
    return x();
}

let myFunction = () => 5*5; // create function
let takeFunValue = doSomething(myFunction); // myFunction exicute insde the doSomthing
console.log(takeFunValue); // 25


```

```
// ********** Example **********
function pow(x, y){
    let total = 1;
    for(let i = 0; i < y; i++){
        total *= x;
    }
    return total;
}
console.log(pow(2, 3)); // 8

// ****** function insert into a variable **********

// that vairable is refer to function
let myFun = function pow(x, y){
    return Math.pow(x, y);
}
console.log(myFun(2, 3)); // 8

// ****** if we insert function into variable then put fun name not mandetory ********
let myFun2 = function (x, y) {
    return Math.pow(x, y);
}
console.log(myFun2(2, 3)); // 8

// =============== Hoisting pratice ===========
// very important concept

    // In function decoration total function is hosted
    // in function expression only vairable is hosted


// not create vaiable x.. shows not define variable
console.log(x); // x is not defined at demo.js:83
// But we create global variable x then CNL show "undefined"
// here create variable x "undefined" show now
var x = 10;

// simple structure like this...
/*
    var x;
    console.log(x); // undefined;
    x = 10; // set value..
*/

// ********* function host first time but call happend second time *****


    doStuff(); // call work.. coz second time it call
// but first time function store
function doStuff (){
    console.log("Mehedi");
}
// simple thik like this
/*
```

```

    function doStuff (){
    console.log("Mehedi");
}
-> then call like normal way
    doStuff()

*/


// ******** But if we create function variable then *********

// "function exprassion"  that means function insert into a variable

// function exprention only variable is hosted or store
//doStuff2();

var doStuff2 = function (){
    console.log("Mehedi");
}

// it will not work .. coz store that way not call before
// if we call After create it will work
/*

let doStuff2 = function (){
    console.log("Mehedi");
}

doStuff2()/

*/


// ================== Function Expression ==========
// when function assign a variable then it is called expression
let myfun3 = function fun(x, y){
    console.log(x + " " + y);
}
// function assing into a varialbe
let myfun4 = myfun3;

myfun4(3, 4); // 3 4

// ******** function assign into a Array *******

let myArray = [myfun3];
console.log(myArray); // 0: Ć fun(x, y)

// call the array function
myArray[0](10, 20); // 10 20 print insde the function

// ********** function use inside the OBJECT **********

```

```javascript
let mathFun = {
    power: myfun3,
};
mathFun.power(30, 40); // 30 40 print inside the function


// =========== Call Back function ===========

// callback means big function exicute small function..
function add(a, b){
    return a + b;
};

function callBackExample(callback){
    return callback(3, 5);
}

console.log(callBackExample(add)); // 8


// =========== return a function from a function ========
function sub(x, y){
    return x - y;
};
function returnFunction(){
    return sub;
};
let result = returnFunction()(30, 10); // call this way
console.log(result); // 20
```