/\*

  call () method :

  \*-The call () method is a predefined JavaScript method.

  \*-It can be used invoke (call) a method with an owner object as an arguments (parameters).

  \*-It can also be used to return a value from a method.

  \*-With call() , an object can use a method belonging to another object.

/\*

/\*

$Example 1:

const person1 = {

    fName: "rakesh",

    lName: "kumar",

    fullName: function(){

        return this.fName+ " " + this.lName;

    }

}

const person2 = {

    fName: "arun",

    lName: "singh",

}

$Function borrowing:

console.log(person1.fullName.call(person2));

 \*/

/\*

$-Example 2:

const person1 = {

    fName: "rakesh",

    lName: "kumar",

    fullName: function(hometown){

        return this.fName+ " " + this.lName + " " + hometown;

    }

}

const person2 = {

    fName: "arun",

    lName: "singh",

}

$-Function borrowing:

console.log(person1.fullName.call(person2));

console.log(person1.fullName.call(person2,"pune"));

 \*/

// apply():

//  -- the apply() method is similar to the call() method.

//  -- the difference:

//  -- the call() method takes arguments seperately.

//  -- the apply() method takes arguments as an array.

// const person1 = {

//     fName: "rakesh",

//     lName: "kumar",

//     fullName: function(hometown,country){

//         return this.fName+ " " + this.lName + " " + hometown + " " + country;

//     }

// }

// const person2 = {

//     fName: "arun",

//     lName: "singh",

// }

// // $-Function borrowing:

// console.log(person1.fullName.call(person2,"indore","india"));

// // call():

// console.log(person1.fullName.call(person2, "pune", "india"));

// // apply():

// console.log(person1.fullName.apply(person2, ["mumbai", "india"]));

// \*-bind():

//          -the blind() method , an object can borrow a method from objects.

//

// const result = person1.fullName.bind(person2,["mumbai", "india"])

// console.log(result); //fullName() will be storedin result varable.

// console.log(result());  // retuen the fullName().

// \* -destructing array:

//          - The Destructing array is a javscript expression that makes it possible to

//            unpack values from arrays, or properties from objects, into distinct variables.

// const arr = [123, "apple", true]

// const [value, fruit, truth] = arr;

// // console.log(fruit);

// console.log(truth);

// const arr = [123, "apple", ,true,["rohit",5]]

// const [value, fruit, truth=500 , opinion , [name , id]] = arr;

// console.log(fruit);

// console.log(id);

// console.log(truth);

// console.log(arr[0]);  //Normal Way.

// console.log(arr[1]);  //Normal Way.

// function calculate(a, b) {

//     const add = a + b;

//     const sub = a - b;

//     const mul = a \* b;

//     return [add, sub, mul];

// }

// const [add, sub, mul] = calculate(4, 5);  //destructing array.

// console.log(add);

// console.log(sub);

// console.log(mul);

//Destructing objects: