Practical:1

Aim: implement following ECMASCRIPT concept

* let, var and const
* destructuring
* map, filter and reduce
* callback, promises and async/wait

A)

Code:

var x = 14;

let y=40;

const z = 15;

x = 12;

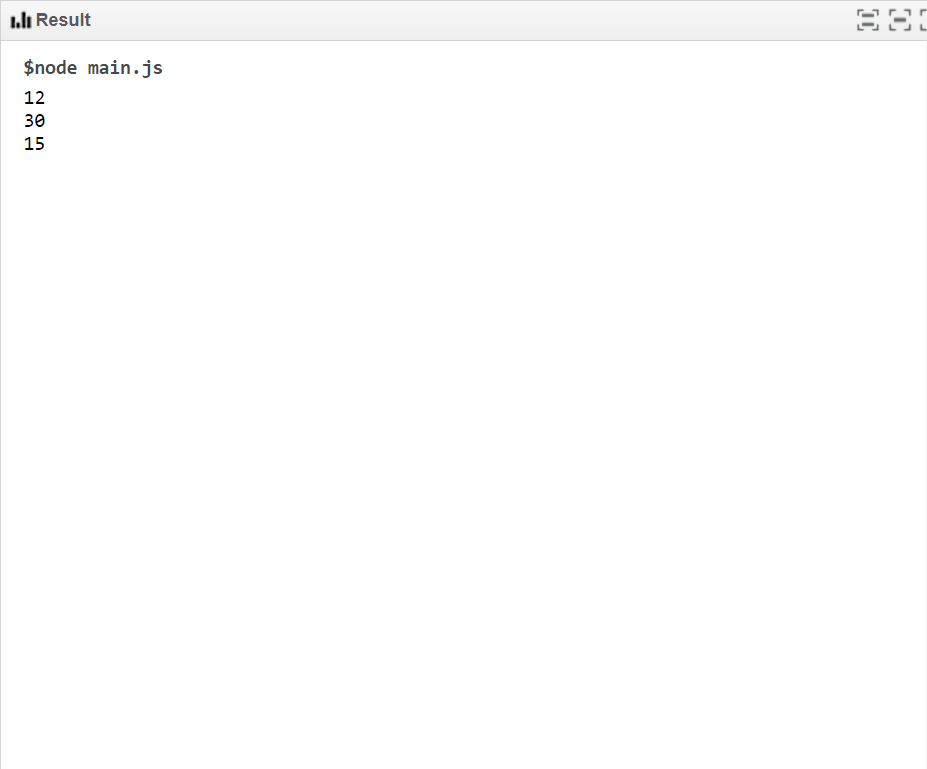
y=30;

console.log(x);

console.log(y);

console.log(z);

output:



B)

function NamesList() {

return ["alpha", "beta", "gamma", "delta"]

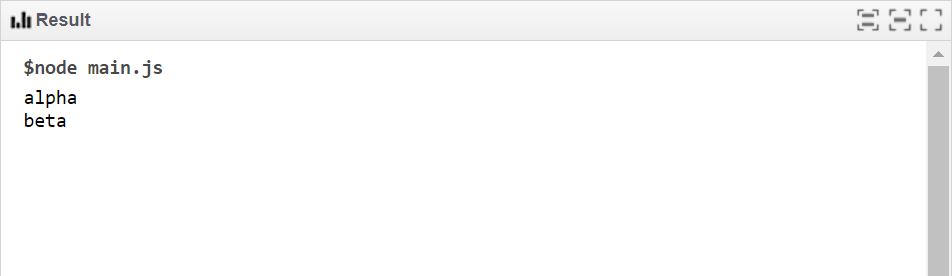
}

var[firstName, secondName] = NamesList();

console.log(firstName);

console.log(secondName);

Output:



c)

const numbers = [1, 2, 3, 4];

const num = [5,6,7,8];

var pets = ['dog', 'chicken', 'cat', 'dog', 'chicken', 'chicken', 'rabbit'];

const doubled = numbers.map(item => item \* 2);

const evens = numbers.filter(item => item % 2 === 0);

var petCounts = pets.reduce(function(obj, pet){

if (!obj[pet]) {

obj[pet] = 1;

} else {

obj[pet]++;

}

return obj;

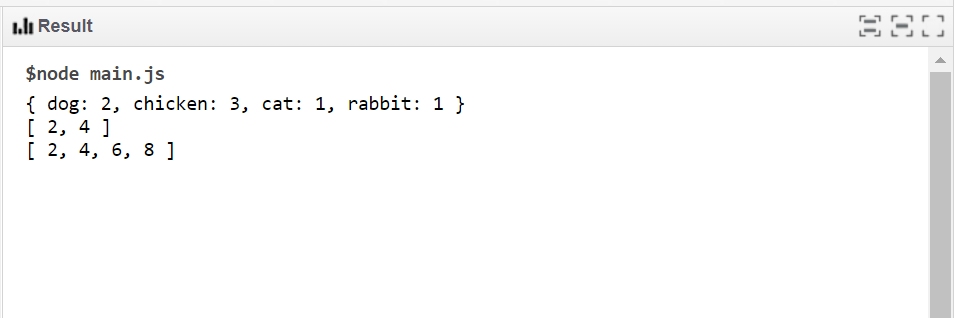
}, {});

console.log(petCounts);

console.log(evens);

console.log(doubled);

output:



D)

//callback

function first() {

console.log(1)

}

function second(callback) {

setTimeout(() => {

console.log(2)

callback()

}, 0)

}

function third() {

console.log(3)

}

first()

second(third)

//promise

function getUsers(onSuccess) {

return new Promise((resolve, reject) => {

setTimeout(() => {

if (onSuccess) {

resolve([

{id: 1, name: 'Jerry'},

{id: 2, name: 'Elaine'},

{id: 3, name: 'George'},

])

} else {

reject('Failed to fetch data!')

}

}, 1000)

})

}

getUsers(false)

.then((response) => {

console.log(response)

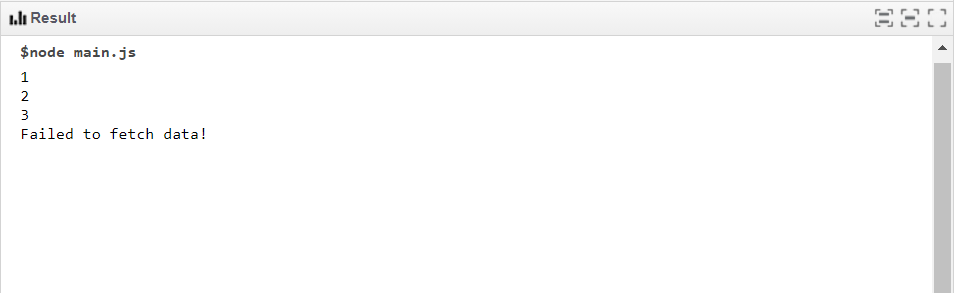
})

.catch((error) => {

console.error(error)

})

Output:



//**Async**

function Sync(){

setTimeout(()=>{

console.log("CSPIT");

},5000);

}

console.log("welcome");

Sync(); //5sec

console.log("charusat");

Output:

