***This code Print the max number in the array***

let marks = [70,40,60,90,45]

let max = marks[0]

for (i=0; i<marks.length; i++){

if(max< marks[i]){

max=marks[i]

}

}

console.log(max);

***This code Print the sum one by one and give final sum***

let marks = [70,40,60,90,45]

let sum = 0

for (i=0; i<marks.length; i++){

sum = sum + marks[i]

console.log(sum);

**Output:**

70

110

170

260

305

***This code Print the sum of values in array***

let marks = [70,40,60,90,45]

let sum = 0

function printMarks(marks){

sum = sum + marks }

marks.forEach(printMarks)

console.log(sum);

##### Output:

305

***This code Print the eligible or not***

var ajith = 26;

function isLegal(age){

if(age>=18){

console.log("Major")

}else{

console.log("Minon")

}

}

isLegal(ajith)

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

let repeat = {};

for(var index in arr){

//console.log(index, arr[index])

var currentValue = arr[index]

//console.log("before", currentValue, repeat)

if(repeat[currentValue]){

repeat[currentValue] +=1

}else{

repeat[currentValue] = 1

} //console.log("after", currentValue, repeat)

}

console.log(repeat);

var array = [1, 2, 5, 2, 4, 4],

nonRepeat = array.filter((v, i) => array.indexOf(v) === array.lastIndexOf(v));

console.log(nonRepeat);

/ Task// temporal dead zone - google doc

Task 3IIFE – WHY we need this AND ADVANMTAGES

**SHADOWING**

Same named variable shadowed the outside the block, both has a same memory location.

var a = 10;

{

var a = 20;

var b = 45;

console.log(a)

console.log(b)

}

console.log(a)

##### Output:

20

45

20

In Let inside the block will be shadowed, but outside the block it will not shadowed,

Because Let is a block scope

let a = 10;

{

let a = 20;

let b = 45;

console.log(a)

console.log(b)

}

console.log(a)

##### Output:

20

45

10

**IIFE**

IIFE - Immediately-invoked Function Expression

IIFE is a JavaScript function that runs as soon as it is defined.

Syntax:

(function name(){

})()

Advantage of IIFE is that any "Function or Variable" defined inside IIFE, cannot be accessed outside the IIFE block, thus preventing global scope from getting polluted. Also helps us manage memory in an efficient manner

**temporal dead zone**

Accessing a var before it is declared has the result undefined

accessing let and const values before they are initialized can cause a ReferenceError because of something called the **temporal dead zone.**

**Use map to double the values**

let arr = [10, 70, 40, 80];

let double = arr => arr \* 2;

let result = arr.map(double);

console.log(result);

[ 20, 140, 80, 160 ]

**Push method**

let arr1 = [10, 20]

arr1.push(50);

console.log(arr1);

[ 10, 20, 50 ]

**Filter method**

var marks = [80, 50, 90, 100, 30, 20, 70]

var pass = marks => marks >= 40;

var res = marks.filter(pass)

console.log(res)

[ 80, 50, 90, 100, 70 ]

To get sum of numbers

const num = [10,20,30,40,50]

console.log(num.reduce((curr, sum)=> sum+curr,0))

150

**const scores = [**

**{**

**marks: 32,**

**name: "Yvette Merritt",**

**},**

**{**

**marks: 57,**

**name: "Lillian Ellis",**

**},**

**{**

**marks: 22,**

**name: "Mccall Carter",**

**},**

**{**

**marks: 21,**

**name: "Pate Collier",**

**},**

**{**

**marks: 91,**

**name: "Debra Beard",**

**},**

**{**

**marks: 75,**

**name: "Nettie Hancock",**

**},**

**{**

**marks: 20,**

**name: "Hatfield Hodge",**

**},**

**];**

**To get Names**

**const val = scores => scores.name**

**const fname= scores.map(val)**

**console.log(fname);**

**To get Passed marks**

**var isPass = scores => scores.marks >= 40;**

**var res = scores.filter(isPass)**

**console.log(res)**

**To get failed students names**

**var failedStudents = scores.filter((score)=> score.marks<=40);**

**var failedNames = failedStudents.map((score)=>scores.name)**

**console.log(scores.filter((score)=>score.marks<=40).map((score)=>score.name))**

**To get marks**

**var ave = scores.map((scores)=>scores.marks)**

**console.log(ave)**

**To get average**

**var ave = scores.map((scores)=>scores.marks)**

**var average =ave.reduce((curr, final)=> curr+final)/ave.length**

**console.log(average.toFixed(2))**

**highest mark name method 1**

**var topper = scores.reduce((top, currStu)=>{**

**if(top.marks< currStu.marks){**

**return currStu;**

**}else {**

**return top;**

**}**

**})**

**console.log(topper.name)**

**highest mark name method 2**

**var topper = scores.reduce((top, currStu)=>**

**top.marks< currStu.marks ? currStu : top)**

**console.log(topper.name)**

**// Task// call, apply, bind eg: with a fn takes arguments**

**Print odd numbers in an array**

let num = [2, 10, 5, 9]

for(i=0; i<num.length; i++){

if(num[i]%2 !=0) {

console.log(num[i])

}

}

Convert all the strings to title caps in a string array

Sum of all numbers in an array