**JAVA SCRIPT**

*JavaScript is the****Programming* Language** for the Web.

JavaScript can update and change both **HTML** and **CSS.**

JavaScript can **calculate**, **manipulate** and **validate** data.

**/\*\*\*\* Section 2👉 we need to do it in console \*\*\*\*/**

**// alert("Welcome, to Complete JavaScript course");**

**// console.log('Welcome, to complete JavaScript Course');**

**/\*\*\*\* Section 3👉 values and variables in JavaScript \*\*\*\*/**

// var myName = 'Sarosh Ahmed';

// var myAge = 26;

// console.log(myage);

// Naming Practice

// var \_myName = "Sarosh";

// var 1myName = "Ahmed";

// var \_1my\_\_Name = "Khan”;

// var $myName = "Sarosh Ahmed";

// var myNaem% = "Sarosh AHmed";

// console.log(myNaem%);

**/\*\*\*\* Section 4👉 Data Types in JavaScript \*\*\*\*/**

// var myName = "Sarosh AHmed";

// console.log(myName);

// var myAge = 26;

// console.log(myAge);

// var iamsarosh = false;

// console.log(isarosh);

// // typeof operator

// console.log(typeof(iAmsarosh));

// DataTypes Practice

// console.log( 10 + "20");

// 9 - "5"

// console.log( 9 - "5"); //bug

// "Java" + "Script"

// console.log( "Java "+ "Script");

// " " + " "

// console.log( " " + 0);

// " " + 0

// "sarosh" - "Ahmed"

// true + true

// true + false

// false + true

// false - true

**// 🙋‍👨‍🏫 Interview Question 1 🙋‍👨‍🏫**

// Difference between null vs undefined?

// var iAmUseless = null;

// console.log(iAmUseless);

// console.log(typeof(iAmUseless));

// //2nd javascript bug

// var iAmStandBy;

// console.log(iAmStandBy);

// console.log(typeof(iAmStandBy));

**// 🙋‍👨‍🏫 Interview Question 2 🙋‍👨‍🏫**

// What is NaN?

// NaN is a property of the global object.

// In other words, it is a variable in global scope.

// The initial value of NaN is Not-A-Number

// var myPhoneNumber = 9876543210;

// var myName = "Sarosh";

// console.log(isNaN(myPhoneNumber));

// console.log(isNaN(myName));

// if(isNaN(myName)){

// console.log("plz enter valid phone no");

// }

**/\*\*\*\* Section 5👉 Arithmetic operators in JavaScript \*\*\*\*/**

// console.log(5+20);

// 1️⃣Assignment operators

// An assignment operator assigns a value to its left operand

// based on the value of its right operand.

// The simple assignment operator is equal (=)

// var x = 5;

// var y = 5;

// console.log("is both the x and y are equal or not" + x == y );

// I will tell you when we will see es6

// console.log(`Is both the x and y are equal : ${x == y}`);

// 2️⃣Arithmetic operators

// An arithmetic operator takes numerical values

// (either literals or variables) as their operands and

// returns a single numerical value.

// console.log(3+3);

// console.log(10-5);

// console.log(20/5);

// console.log(5\*6);

// console.log("Remainder Operator " + 27%4);

// 🙄Increment and Decrement operator

// Operator: x++ or ++x or x-- or --x

// If used postfix, with operator after operand (for example, x++),

// the increment operator increments and returns the value before incrementing.

// var num = 15;

// var newNum = num-- + 5;

// console.log(num);

// console.log(newNum);

// Postfix increment operator means the expression is evaluated

// first using the original value of the variable and then the

// variable is incremented(increased).

// If used prefix, with operator before operand (for example, ++x),

// the increment operator increments and returns the value after incrementing.

// var num = 15;

// var newNum = --num + 5;

// console.log(num);

// console.log(newNum);

// Prefix increment operator means the variable is incremented first and then

// the expression is evaluated using the new value of the variable.

// 3️⃣Comparison operators

// A comparison operator compares its operands and

// returns a logical value based on whether the comparison is true.

// var a = 30;

// var b = 10;

// Equal (==)

// console.log(a == b);

// Not equal (!=)

// console.log(a != b);

// // Greater than (>)

// console.log(a > b);

// // Greater than or equal (>=)

// console.log(a >= b);

// // Less than (<)

// console.log(a < b);

// // Less than or equal (<=)

// console.log(a <= b);

// 4️⃣ Logical operators

// Logical operators are typically used with Boolean (logical) values;

// when they are, they return a Boolean value.

// var a = 30;

// var b = -20;

// Logical AND (&&)

// The logical AND (&&) operator (logical conjunction) for a set of

// operands is true if and only if all of its operands are true.

// console.log(a > b && b > -50 && b < 0);

// Logical OR (||)

// The logical OR (||) operator (logical disjunction) for a set of

// operands is true if and only if one or more of its operands is true.

// console.log((a < b) || (b > 0) || (b > 0));

// Logical NOT (!)

// The logical NOT (!) operator (logical complement, negation)

// takes truth to falsity and vice versa.

// console.log(!((a>0) || (b<0)));

// console.log(!true);

// 5️⃣ String Concatenation(operators)

// The concatenation operator (+) concatenates two string values together,

// returning another string that is the union of the two operand strings.

// console.log("Hello World");

// console.log("hello " + "world");

// var myName = "Sarosh";

// console.log(myName + " Khan");

// console.log(myName + " Ahmed");

// console.log(myName + " Khanzada");

// 😳 4 Challenge Time

// What will be the output of 3\*\*3?

// What will be the output, when we add a number and a string?

// Write a program to swap two numbers?

// Write a program to swap two numbers without using third variable?

// sol 1: ✔

// console.log(9\*\*2); // 9\*9

// console.log(10 \*\* -1); 1/10

// sol 2: ✔

// console.log(5 + "");

// sol 3: ✔

// var a = 5;

// var b = 10;

// output b=5; a=10

// var c = b; //c = 10

// b = a; // b = 5;

// a = c;

// console.log("the value of a is " + a);

// console.log("the value of b is " + b);

// sol 4: ✔

// var a = 5;

// var b = 10;

// // output b=5; a=10

// a = a + b; // a = 15

// b = a - b; // b = 5;

// a = a - b; // a = 10;

// console.log("the value of a is " + a);

// console.log("the value of b is " + b);

// 🙋‍👨‍🏫 Interview Question 4 🙋‍👨‍🏫

// What is the Difference between == vs === ?

// sol

// var num1 = 5;

// var num2 = '5';

// console.log(typeof(num1));

// console.log(typeof(num2));

// console.log(num1 == num2 );

// var num1 = 5;

// var num2 = '5';

// console.log(typeof(num1));

// console.log(typeof(num2));

// console.log(num2);

// console.log(num1 === num2 );

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**/\*\*\*\* Section 6👉 Control Statement -**

\*

\* 1️⃣If...Else \*/

// The if statement executes a statement if a specified condition is truthy.

// If the condition is falsy, another statement can be executed.

// if raining = raincoat

// else no raincoat

// var tomr = 'sunny';

// if(tomr == 'rain'){

// console.log('take a raincoat');

// }else{

// console.log('No need to take a raincoat');

// }

// 2️⃣Conditional (ternary) operator

// The conditional (ternary) operator is the only JavaScript operator

// that takes three operands

// var age = 17;

// if(age >= 18){

// console.log("you are eligible to vote");

// }else{

// console.log("you are not eligible to vote");

// }

// var age = 18;

// console.log((age >= 18) ? "you can vote" : "you can't vote");

// 3️⃣ switch Statement

// Evaluates an expression, matching the expression's value to a

// case clause, and executes statements associated with that case.

// 1st without break statment

// Find the Area of circle, triangle and rectangle?

// var area = "square" ;

// var PI = 3.142, l=5, b=4, r=3;

// if(area == "circle"){

// console.log("the area of the circle is : " + PI\*r\*\*2);

// }else if(area == "triangle"){

// console.log("the area of the triangle is : " + (l\*b)/2);

// }else if(area == "rectangle"){

// console.log("the area of the rectangle is : " + (l\*b));

// }else{

// console.log("please enter valid data");

// }

// var area = "dsfsad" ;

// var PI = 3.142, l=5, b=4, r=3;

// switch(area){

// case 'circle':

// console.log("the area of the circle is : " + PI\*r\*\*2);

// break;

// case 'triangle':

// console.log("the area of the triangle is : " + (l\*b)/2);

// break;

// case 'rectangle':

// console.log("the area of the rectangle is : " + (l\*b));

// break;

// default:

// console.log("please enter valid data");

// }

// 🤗break

// Terminates the current loop, switch, or label

// statement and transfers

// program control to the statement following the terminated statement.

// 🤗continue

// Terminates execution of the statements in the current iteration of the

// current or labeled loop, and continues execution of the loop with the

// next iteration.

// 4️⃣ While Loop Statement

// The while statement creates a loop that executes a specified statement

// as long as the test condition evaluates to true.

// var num=20;

// // block scope

// while(num <= 10){

// console.log(num); //infinte loop

// num++;

// }

// 5️⃣ Do-While Loop Statement

// var num = 20;

// do{

// debugger;

// console.log(num); //infinte loop

// num++;

// }while(num <= 10);

// 6️⃣ For Loop

// for(var num = 0; num <= 10; num++){

// debugger;

// console.log(num);

// }

// 😀6: challenge Time 🏁

// JavaScript program to print table for given number (8)?

// output : 8 \* 1 = 8

// 8 \* 2 = 16(8\*2)

// => 8 \* 10 = 80

// for(var num = 1; num<= 10; num++){

// var tableOf = 12;

// console.log(tableOf + " \* " + num + " = " + tableOf \* num);

// }

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**/\*\*\*\* Section 5 👉 Functions in JavaScript \*\*\*\*/**

// A JavaScript function is a block of code designed to perform a particular task.

// 1️⃣Function Definition

// Before we use a function, we need to define it.

// A function definition (also called a function declaration, or function statement)

// consists of the function keyword, followed by:

// The name of the function.

// A list of parameters to the function, enclosed in parentheses and separated by commas.

// The JavaScript statements that define the function, enclosed in curly brackets, {...}.

// var a = 10;

// var b = 20;

// var sum = a+b;

// console.log(sum);

// function sum(){

// var a = 10, b = 40;

// var total = a+b;

// console.log(total);

// }

// //

// 2️⃣Calling functions

// Defining a function does not execute it.

// A JavaScript function is executed when "something" invokes it (calls it).

// function sum(){

// var a = 10, b = 40;

// var total = a+b;

// console.log(total);

// }

// sum();

// 3️⃣ Function Parameter vs Function Arguments

// Function parameters are the names listed in the function's definition.

// Function arguments are the real values passed to the function.

// function sum(a,b){

// var total = a+b;

// console.log(total);

// }

// sum();

// sum(20,30);

// sum(50,50);

// sum(5,6)

// // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 🙋‍👨‍🏫 Interview Question 🙋‍👨‍🏫

// Why Functions?

// You can reuse code: Define the code once, and use it many times.

// You can use the same code many times with different arguments,

// to produce different results.

// OR

// A function is a group of reusable code which can be called anywhere

// in your program. This eliminates the need of writing the same code

// again and again.

// DRY => do not repeat yourself

// 4️⃣ Function expressions

// "Function expressions simply means

// create a function and put it into the variable "

// function sum(a,b){

// var total = a+b;

// console.log(total);

// }

// var funExp = sum(5,15);

// 5️⃣ Return Keyword

// When JavaScript reaches a return statement,

// the function will stop executing.

// Functions often compute a return value.

// The return value is "returned" back to the "caller"

// function sum(a,b){

// return total = a+b;

// }

// var funExp = sum(5,25);

// console.log('the sum of two no is ' + funExp );

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 👻 Now It's Time for Modern JavaScript 😍😍

// 🙏🙏 Features of ECMAScript 2015 also known as ES6 🙏🙏

// 1️⃣ LET VS CONST vs VAR

// var myName = "Kahyan Khan";

// console.log(myName);

// myName = "Sarosh Ahmed";

// console.log(myName);

// let myName = "Khanzada Rehman";

// console.log(myName);

// myName = "Aziz Bhai Ceo";

// console.log(myName);

// const myName = "Sarosh AHmed";

// console.log(myName);

// myName = "Kamran";

// console.log(myName);

// function biodata() {

// const myFirstName = "";

// console.log(myFirstName);

// if(true){

// const myLastName = "Ahmed";

// }

// // console.log('innerOuter ' + myLastName);

// }

// console.log(myFirstName);

// biodata();

// var => Function scope

// let and const => Block Scope

// 2️⃣ Template literals (Template strings)

// JavaScript program to print table for given number (8)?

// output : 8 \* 1 = 8

// 8 \* 2 = 16(8\*2)

// => 8 \* 10 = 80

// for(let num = 1; num<= 10; num++){

// let tableOf = 12;

// // console.log(tableOf + " \* " + num + " = " + tableOf \* num);

// console.log( ` ${tableOf} \* ${num} = ${tableOf \* num}` );

// }

// 3️⃣ Default Parameters

// Default function parameters allow named parameters to be

// initialized with default values if no value or undefined is passed.

// function mult(a,b=5){

// return a\*b;

// }

// console.log(mult(3));

// 4️⃣ Destructuring in ES6

// The destructuring assignment syntax is a JavaScript expression

// that makes it possible to unpack values from arrays,

// or properties from objects, into distinct variables.

// ➡ Array Destructuring 🏁

// const myBioData = ['Sarosh', 'Ahmed', 26];

// let myFName = myBioData[0];

// let myLName = myBioData[1];

// let myAge = myBioData[2];

// let [myFName,myAge, myLName] = myBioData;

// console.log(myAge);

// we can add values too

// let [myFName,myLName,myAge, myDegree="MCS"] = myBioData;

// console.log(myDegree);

// ➡ Object destructuring 🏁

// const myBioData = {

// myFname : 'Sarosh',

// myLname : 'Ahmed',

// myAge : 26

// }

// let age = myBioData.age;

// let myFname = myBioData.myFname;

// let {myFname,myLname,myAge, myDegree="MCS"} = myBioData;

// console.log(myLname);

// 5️⃣ Object Properties

// ➡ we can now use Dynamic Properties

// let myName = "Sarosh";

// const myBio = {

// [myName] : "hello how are you?",

// [20 + 6] : "is my age"

// }

// console.log(myBio);

// ➡ no need to write key and value, if both are same

// let myName = "Sarosh Ahmed";

// let myAge = 26;

// const myBio = {myName,myAge}

// console.log(myBio);

// 6️⃣ Fat Arror Function

// 👻 Normal Way of writing Function

// console.log(sum());

// function sum() {

// let a = 5; b = 6;

// let sum = a+b;

// return `the sum of the two number is ${sum}`;

// }

// 👻 How to convert in into Fat Arrow Function

// const sum = () => `the sum of the two number is ${(a=5)+(b=6)}`;

// console.log(sum());

// 7️⃣Spread Operator

// const colors = ['red', 'green', 'blue', 'white', 'pink'];

// const myColors = ['red', 'green', 'blue', 'white','pink', 'yellow', 'black'];

// // // 2nd time add one more color on top and tell we need to write it again

// // // on myColor array too

// const MyFavColors = [ ...colors, 'yellow', 'black'];

// console.log(MyFavColors);

// ES7 features

// 1: array include

// const colors = ['red', 'green', 'blue', 'white', 'pink'];

// const isPresent = colors.includes('purple');

// console.log(isPresent);

// 2: \*\*

// console.log(2\*\*3);

// ES8 Features

// String padding

// Object.values()

// Object.entries()

// const message = "my name is Kahyan";

// console.log(message);

// console.log(message.padStart(5));

// console.log(message.padEnd(10));

// const person = { name: 'Fred', age: 87 };

// // // console.log( Object.values(person) );

// const arrObj = Object.entries(person);

// console.log(Object.fromEntries(arrObj));

// ES2018

// const person = { name: 'Fred', age: 87, degree : "mcs" };

// const sPerson = { ...person };

// console.log(person);

// console.log(sPerson);

// ES2019

// Array.prototype.{flat,flatMap}

// Object.fromEntries()

// ES2020

// #1: BigInt

// let oldNum = Number.MAX\_SAFE\_INTEGER;

// // console.log(oldNum);

// // console.log( 9007199254740991n + 12n );

// const newNum = 9007199254740991n + 12n;

// console.log(newNum);

// console.log(typeof newNum);

// const foo = null ?? 'default string';

// console.log(foo);

// ES2014

// "use strict";

// x = 3.14;

// console.log(x);

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*\*\* Section 7👉 Arrays in JavaScript \*\*\*\*/

// When we use var, we can stored only one value at a time.

// var friend1 = 'ramesh';

// var friend2 = 'arjun';

// var friend3 = 'vishal';

// var myFriends = ['ramesh',22,male,'arjun',20,male,'vishal',true, 52];

// When we feel like storing multiple values in one variable then

// instead of var, we will use an Array.

// In JavaScript, we have an Array class, and

// arrays are the prototype of this class.

// example 🏁

// var myFriends = ['ramesh',22,male,'arjun',20,male,'vishal',true, 52];

// 1️⃣ Array Subsection 1 👉 Traversal in array✌

// navigate through an array

// if we want to get the single data at a time and also

// if we want to change the data

// var myFriends = ['Sarosh','Muzammil','Idrees','Hamza'];

// console.log(myFriends[myFriends.length - 1]);

// if we want to check the length of elements of an array

// console.log(myFriends.length);

// we use for loop to navigate

var myFriends = ['Sarosh','Muzammil','Idrees','Hamza'];// for(var i=0; i<myFriends.length; i++){

// console.log(myFriends[i]);

// }

// After ES6 we have for..in and for..of loop too

// var myFriends = ['Sarosh','Muzammil','Idrees','Hamza'];

// for(let elements in myFriends){

// console.log(elements);

// }

// for(let elements of myFriends){

// console.log(elements);

// }

// Array.prototype.forEach() 🙋‍♂️

// Calls a function for each element in the array.

// var myFriends = ['Sarosh','Muzammil','Idrees','Hamza'];

// myFriends.forEach(function(element, index, array) {

// console.log(element + " index : " +

// index + " " + array);

// });

// myFriends.forEach((element, index, array) => {

// console.log(element + " index : " +

// index + " " + array);

// });

// 2️⃣ Array Subsection 2 👉 Searching and Filter in an Array

// Array.prototype.indexOf() 🙋‍♂️

// Returns the first (least) index of an element within the array equal

// to an element, or -1 if none is found. It search the element from the

// 0th index number

// var myFriendNames = ["Sarosh","Ahmed","Khan","Aziz","Kahyan"];

// console.log(myFriendNames.indexOf("Sarosh", 3));

// Array.prototype.lastIndexOf() 🙋‍♂️

// Returns the last (greatest) index of an element within the array equal

// to an element, or -1 if none is found. It search the element last to first

// var myFriendNames = ["Sarosh","Ahmed","Khan","Aziz","Kahyan"];

// console.log(myFriendNames.lastIndexOf("aziz",3));

// Array.prototype.includes() 🙋‍♂️

// Determines whether the array contains a value,

// returning true or false as appropriate.

// var myFriendNames = ["Sarosh","Ahmed","Khan","Aziz","Kahyan"];

// console.log(myFriendNames.includes("Ali"));

// Array.prototype.find() 🙋‍♂️

// arr.find(callback(element[, index[, array]])[, thisArg])

// Returns the found element in the array, if some element in the

// array satisfies the testing function, or undefined if not found.

// Only problem is that it return only one element

// const prices = [200,300,350,400,450,500,600];

// price < 400

// const findElem = prices.find((currVal) => currVal < 400 );

// console.log(findElem);

// console.log( prices.find((currVal) => currVal > 1400 ) );

// Array.prototype.findIndex() 🙋‍♂️

// Returns the found index in the array, if an element in the

// array satisfies the testing function, or -1 if not found.

// console.log( prices.findIndex((currVal) => currVal > 1400 ) );

// Array.prototype.filter() 🙋‍♂️

// Returns a new array containing all elements of the calling

// array for which the provided filtering function returns true.

// const prices = [200,300,350,400,450,500,600];

// // price < 400

// const newPriceTag = prices.filter((elem, index) => {

// return elem > 1400;

// })

// console.log(newPriceTag);

// 3️⃣ Array Subsection 3 👉 How to sort an Array

// Array.prototype.sort() 🙋‍♂️

// The sort() method sorts the elements of an array in place and

//returns the sorted array. The default sort order is ascending, built

//upon converting the elements into strings,

// then comparing their sequences of UTF-16 code units values.

// const months = ['March', 'Jan', 'Feb', 'April' , 'Dec', 'Nov'];

// console.log(months.sort());

// const array1 = [1, 30, 4, 21, 100000, 99];

// console.log(array1.sort());

// However, if numbers are sorted as strings,

// "25" is bigger than "100", because "2" is bigger than "1".

// Because of this, the sort() method will produce an incorrect

// result when sorting numbers.

// 2: sort the array in descending order

// var fruits = ["Banana", "Orange", "Apple", "Mango"];

// let aFruits = fruits.sort();

// //Array.prototype.reverse() 🙋‍♂️

// // The reverse() method reverses an array in place.

// // The first array element becomes the last, and

// // the last array element becomes the first.

// 4️⃣ Array Subsection 4 👉 Perform CRUD

// Array.prototype.push() 🙋‍♂️

// The push() method adds one or more elements to the

// end of an array and returns the new length of the array.

// const animals = ['pigs', 'goats', 'sheep'];

// // const count = animals.push('chicken');

// // console.log(count);

// animals.push('chicken', 'cats','cow');

// console.log(animals);

// Array.prototype.unshift() 🙋‍♂️

// The unshift() method adds one or more elements to the

// beginning of an array and returns the new length of the array.

// const animals = ['pigs', 'goats', 'sheep'];

// const count = animals.unshift('chicken');

// console.log(count);

// console.log(animals);

// animals.unshift('chicken', 'cats','cow');

// console.log(animals);

// 2nd example

// const myNumbers = [1,2,3,5];

// myNumbers.unshift(4,6);

// console.log(myNumbers);

// Array.prototype.pop() 🙋‍♂️

// The pop() method removes the last element from an array and returns

// that element. This method changes the length of the array.

// const plants = ['broccoli', 'cauliflower', 'kale', 'tomato', 'cabbage'];

// console.log(plants);

// console.log(plants.pop());

// console.log(plants);

// Array.prototype.shift() 🙋‍♂️

// The shift() method removes the first element from an array and returns

// that removed element. This method changes the length of the array.

// const plants = ['broccoli', 'cauliflower', 'kale', 'tomato', 'cabbage'];

// console.log(plants);

// console.log(plants.shift());

// console.log(plants);

// 😀8: challenge Time 🏁

// Array.prototype.splice() 🙋‍♂️

// Adds and/or removes elements from an array.

// 1: Add Dec at the end of an array?

// 2: What is the return value of splice method?

// 3: update march to March (update)?

// 4: Delete June from an array?

// sol1:

// const newMonth = months.splice(months.length,0,"Dec");

// console.log(months);

// sol2:

// console.log(newMonth);

// sol3:

// const months = ['Jan', 'march', 'April', 'June', 'July'];

// const indexOfMonth = months.indexOf('June');

// if(indexOfMonth != -1){

// const updateMonth = months.splice(indexOfMonth,1,'june');

// console.log(months);

// }else{

// console.log('No such data found');

// }

// sol3:

// const months = ['Jan', 'march', 'April', 'June', 'July'];

// const indexOfMonth = months.indexOf('April');

// if(indexOfMonth != -1){

// const updateMonth = months.splice(indexOfMonth,2);

// console.log(months);

// console.log(updateMonth);

// }else{

// console.log('No such data found');

// }

// 5️⃣ Array Subsection 4 👉 Map and Reduce Method

// Array.prototype.map() 🙋‍♂️

// let newArray = arr.map(callback(currentValue[, index[, array]]) {

// // return element for newArray, after executing something

// }[, thisArg]);

// Returns a new array containing the results of calling a

// function on every element in this array.

// const array1 = [1, 4, 9, 16, 25];

// num > 9

// let newArr = array1.map((curElem,index,arr) => {

// return curElem > 9;

// })

// console.log(array1);

// console.log(newArr);

// let newArr = array1.map((curElm, index, arr) => {

// return `Index no = ${index} and the value is ${curElm} belong to ${arr} `

// }).reduce().

// console.log(newArr);

// let newArrfor = array1.forEach((curElm, index, arr) => {

// return `Index no = ${index} and the value is ${curElm} belong to ${arr} `

// })

// console.log(newArrfor);

// It return new array without mutating the orignal array

// Array.prototype.find() 🙋‍♂️

// arr.find(callback(element[, index[, array]])[, thisArg])

// Returns the found element in the array, if some element in the

// array satisfies the testing function, or undefined if not found.

// Only problem is that it return only one element

// const prices = [200,300,350,400,450,500,600];

// price < 400

// const findElem = prices.find((currVal) => currVal < 400 );

// console.log(findElem);

// console.log( prices.find((currVal) => currVal > 1400 ) );

**/\*\*\*\* Section 7👉 Strings in JavaScript \*\*\*\*/**