JavaScript Beginning to Mastery Syllabus

* JavaScript vs EcmaScript
* Install vscode and run ﬁrst program

# Basics of Programming in JavaScript

* Hello world program
* // *console.log can print something on console*
* *console.log*("hello world");
* Declare variable using var
* "use strict";
* // *intro to variables*
* // *variables can store some information*
* // *we can use that information later*
* // *we can change that information later*
* // *declare a variable*
* varfirstName="Harshit";
* //*var firstname=' anupam' is possible*
* // *use a variable*
* *console.log*(firstName);
* // *change value*
* firstName = "Mohit";
* *console.log*(firstName);
* More about variable
* // *rules for naming variables*
* // *you cannot start with number*
* // *example :-*
* // *1value (invalid)*
* // *value1 (valid)*
* varvalue1=2;
* *console.log*(value1);
* // *you can use only undersore \_ or dollar symbol*
* // *first\_name (valid)*
* // *\_firstname (valid)*
* // *first$name (valid)*
* // *$firstname (valid)*
* // *you cannot use spaces*
* // *var first\_name = "harshit"; // snake case writing*
* // *var firstName = "harshit"; // camel case writing*
* // *first name (invalid)*
* // *convention*
* // *start with small letter and use camelCase*
* Let
* // *let keyword*
* // *declare variable with let keyword*
* // *let firstName = "harshit";*
* // *firstName = "Mohit";*
* // *console.log(firstName);*
* // *block scope vs funtion scope (covered later in this video)*
* // *let                                                     var*
* // *let is block-scoped.                                        var is function scoped.*
* // *let does not allow to redeclare variables.                   var allows to redeclare variables.*
* // *Hoisting does not occur in let.                             Hoisting occurs in var.*
* // *{*
* // *var a = 34;*
* // *console.log(a);*
* // *}*
* // *console.log(a); value of a can be accessible*
* //*output :*
* // *34*
* // *34*
* // *let name ='anupam';*
* // *let name='juni';*
* // *name can't be initialiseda again*
* // *{*
* // *let a = 34;*
* // *console.log(a);*
* // *}*
* // *console.log(a); value of a can't be accessible*
* // *output: 34*
* // *e: \#Coding playground\3.web dev\3. js notes\#mastery code\javascript - beginning - to - mastery - main\part1\04.js: 26*
* // *console.log(a); //error*
* Const
* // *declare constants*
* const *pi* =3.14;
* *console.log*(pi);
* String Indexing
* // *String indexing*
* letfirstName="harshitdfjakldsfdf";
* // *h    a   r   s   h   i   t*
* // *0    1   2   3   4   5   6*
* // *console.log(firstName[0]);*
* // *length of string*
* // *firstName.length*
* *console.log*(*firstName.*length);
* *console.log*(firstName[*firstName.*length-2]);
* // *last Index : length - 1*
* Useful string methods
* // *trim()*
* // *toUpperCase()*
* // *toLowerCase()*
* // *slice(startIndex, endIndex)//0,1,2,3,4(0,5)*
* // *lastIndexof("word")*
* //*indexof()*
* //*charAt*
* // *endswith*
* // *includes*
* // *substring*
* // *split*
* // *replace*
* letfirstName="harshit";
* // *console.log(firstName.length);*
* // *firstName ="    aafasd   "*
* // *firstName = firstName.trim(); // "harshit"*
* // *console.log(firstName)*
* // *console.log(firstName.length);*
* // *firstName = firstName.toUpperCase();*
* // *firstName = firstName.toLowerCase();*
* // *console.log(firstName);*
* // *start index*
* // *end index*
* letnewString= *firstName.slice*(1); // *hars*
* *console.log*(newString);
* // *typeof operator*
* // *data types (primitive data types)*
* // *string "harhit"*
* // *number 2, 4, 5.6*
* // *booleans*
* // *undefined*
* // *null*
* // *BigInt*
* // *Symbol*
* // *let age = 22;*
* // *let firstName = "harshit";*
* // *// console.log(typeof age);*
* // *// 22 -> "22"*
* // *// convert number to string.*
* // *age = age + "";             ----------add "" to the number*
* // *console.log(typeof(age)); "22" string*
* // *// convert string to number.*
* // *let myStr = +"34"; ---------add + to the string*
* // *console.log(typeof myStr); number*
* // *let age = "18";*
* // *age = Number(age);*
* // *console.log(typeof age);*
* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
* // *string concatenation*
* letstring1="17";
* letstring2="10";
* letnewString=+string1++string2; // *27*
* newString = string1 + string2; // *1710*
* *console.log*(typeof newString);
* // *add + sign in front of string to convert it into the number*
* Template Strings
* // *template string*
* letage=22;
* letfirstName="harshit"
* // *"my name is harshit and my age is 22 "*
* // *let aboutMe = "my name is " + firstName + " and my age is " + age;*
* letaboutMe=`*my name is* ${firstName} *and my age is* ${age}`
* *console.log*(aboutMe);
* Null, undeﬁned, BigInt, typeof
* // *undefined*
* // *null*
* // *let firstName;*
* // *console.log(typeof firstName); output: undefined*
* // *firstName = "Harshit";*
* // *console.log(typeof firstName, firstName); output: string Harshit*
* // *let myVariable = null;*
* // *console.log(myVariable); output:null*
* // *myVariable = "harshit";*
* // *console.log(myVariable, typeof myVariable);*
* // *console.log(typeof null); output:object*
* // *bug , error \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**
* // *BigInt*
* // *let myNumber = BigInt(12);*
* // *let sameMyNumber = 123n; //shorthand add n after number ends*
* // *// console.log(myNumber);*
* // *// console.log(Number.MAX\_SAFE\_INTEGER);//to check the higheest number of value can be added*
* // *console.log(myNumber+ sameMyNumber); only number with same data type can be added like bigInt with bigInt*
* Booleans and Comparison Operator
* // *booleans & comparison operator*
* // *booleans*
* // *true, false*
* // *let num1 = 7;*
* // *let num2 = "7";*
* // *console.log(num1<num2); false*
* // *== vs ===  //== check only for the vale // where === check for value with datatype*
* // *console.log(num1 === num2);*
* // *!=        vs             !==*
* //*with data --------- with data and datatype*
* // *console.log(num1 !== num2);*
* Truthy and Falsy Values
* // *truthy and falsy values*
* // *truthy*
* // *"abc"*
* // *1, -1*
* // *falsy values*
* // *""*
* // *null*
* // *undefined*
* // *0*
* // *false*
* If else statement
* // *if else condition*
* // *let age = 17;*
* // *if(age>=18){*
* // *console.log("User can play ddlc");*
* // *}else {*
* // *console.log("User can play mario");*
* // *}*
* // *let num = 13;*
* // *if(num%2===0){*
* // *console.log("even");*
* // *}else{*
* // *console.log("odd");*
* // *}*
* // *falsy values*
* // *false*
* // *""*
* // *null*
* // *undefined*
* // *0*
* // *truthy*
* // *"abc"*
* // *1, -1*
* // *let firstName= 0;*
* // *if(firstName){*
* // *console.log(firstName);*
* // *}else{*
* // *console.log("firstName is kinda empty");*
* // *}*
* Ternary Operator
* // *ternary operator*
* // *let age = 4;*
* // *let drink;*
* // *if(age>=5){*
* // *drink = "coffee";*
* // *}else{*
* // *drink = "milk";*
* // *}*
* // *console.log(drink);*
* // *ternary operator / conditional operator*
* // *let age = 3;*
* // *let drink = age >= 5 ? "coffee" : "milk";*
* // *console.log(drink);*
* && || operator
* // *and  or operator*
* //*and && when both condition is true*
* //*or || when any of the condition is true*
* // *if(firstName[0] === "H"){*
* // *console.log("your name starts with H")*
* // *}*
* // *if(age > 18){*
* // *console.log("you are above 18");*
* // *}*
* // *if(firstName[0] === "H" && age>18){*
* // *console.log("Name starts with H and above 18");*
* // *}else{*
* // *console.log("inside else");*
* // *}*
* letfirstName="arshit";
* letage=16;
* if (firstName[0] === "H" || age > 18) {
* *console.log*("inside if");
* } else {
* *console.log*("inside else");
* }
* Nested if else
* // *nested if else*
* // *winning number 19*
* // *19 your guess is right*
* // *17 too low*
* // *20 too high*
* letwinningNumber=19;
* letuserGuess=+*prompt*("Guess a number");
* if(userGuess === winningNumber){
* *console.log*("Your guess is right!!");
* }else{
* if(userGuess < winningNumber){
* *console.log*("too low !!!");
* }else{
* *console.log*("too high !!!");
* }
* }
* If elseif else
* // *if*
* // *else if*
* // *else if*
* // *else if*
* // *else*
* // *let tempInDegree = 50;*
* // *if(tempInDegree < 0){*
* // *console.log("extremely cold outside");*
* // *}else if(tempInDegree < 16){*
* // *console.log("It is cold outside");*
* // *}else if(tempInDegree < 25){*
* // *console.log("wheather is okay ");*
* // *}else if(tempInDegree < 35){*
* // *console.log("lets go for swim");*
* // *}else if(tempInDegree < 45){*
* // *console.log("turn on AC");*
* // *}else{*
* // *console.log("too hot!!");*
* // *}*
* // *console.log("hello");*
* // *let tempInDegree = 50;*
* // *if(tempInDegree < 0){*
* // *console.log("extremely cold outside");*
* // *}else if(tempInDegree < 16){*
* // *console.log("It is cold outside");*
* // *}else if(tempInDegree < 25){*
* // *console.log("wheather is okay ");*
* // *}else if(tempInDegree < 35){*
* // *console.log("lets go for swim");*
* // *}else if(tempInDegree < 45){*
* // *console.log("turn on AC");*
* // *}else{*
* // *console.log("too hot!!");*
* // *}*
* // *console.log("hello there");*
* lettempInDegree=4;
* if (tempInDegree > 40) {
* *console.log*("too hot");
* } else if (tempInDegree > 30) {
* *console.log*("lets go for swim");
* } else if (tempInDegree > 20) {
* *console.log*("weather is cool");
* } else if (tempInDegree > 10) {
* *console.log*("it is very cold outside");
* } else {
* *console.log*("extremely cold");
* }
* *console.log*("hello");
* Switch statement
* // *switch statement*
* // *let day = 7;*
* // *if(day === 0){*
* // *console.log("Sunday");*
* // *}else if(day ===1){*
* // *console.log("Monday");*
* // *}else if(day ===2){*
* // *console.log("Tuesday");*
* // *}else if(day ===3){*
* // *console.log("Wednesday");*
* // *}else if(day ===4){*
* // *console.log("Thrusday");*
* // *}else if(day ===5){*
* // *console.log("Friday");*
* // *}else if(day ===6){*
* // *console.log("Saturday");*
* // *}else{*
* // *console.log("Invalid Day");*
* // *}*
* letday=9;
* switch(day){
* case 0:
* *console.log*("Sunday");
* *break*;
* case 1:
* *console.log*("Monday");
* *break*;
* case 2:
* *console.log*("Tuesday");
* *break*;
* case 3:
* *console.log*("Wednesday");
* *break*;
* case 4:
* *console.log*("Thrusday");
* *break*;
* case 5:
* *console.log*("Friday");
* *break*;
* case 6:
* *console.log*("Saturday");
* *break*;
* default:
* *console.log*("Invalid Day");
* }
* While loop
* // *while loop*
* // *0 se 9*
* // *dry don't repeat yourself*
* leti=0; // *1 2 3 4*
* *while*(i<=9){
* *console.log*(i);
* i++;
* }
* *console.log*(`current value of i is ${i}`);
* *console.log*("hello");
* While loop examples
* // *while loop example*
* letnum=100;
* // *let total = 0; //1 + 2 +3*
* // *let i = 0;*
* // *while(i<=100){*
* // *total = total + i;*
* // *i++;*
* // *}*
* // *console.log(total);*
* // *let total = (num\*(num+1))/2;*
* // *console.log(total);*
* For loop
* // *intro to for loop*
* // *print 0 to 9*
* *for*(leti=0;i<=9;i++){
* *console.log*(i);
* }
* // *console.log("value of i is ",i);*
* For loop examples
* // *for loop example*
* lettotal=0;
* letnum=100;
* *for*(leti=1; i<=num; i++){
* total = total + i;
* }
* *console.log*(total);
* Break and continue keyword
* // *break keywork*
* // *continue keyword*
* // *for(let i = 1; i<=10; i++){*
* // *if(i===4){*
* // *break;*
* // *}*
* // *console.log(i);*
* // *}*
* *for* (leti=1; i <= 10; i++) {
* if (i === 4) {
* *continue*;// *skip this iteration*
* }
* *console.log*(i);
* }
* // *1*
* // *2*
* // *3*
* // *5*
* // *6*
* // *7*
* // *8*
* // *9*
* // *10*
* // *console.log("hello there");*
* Do while loop
* // *do while loop*
* // *while(i<=9){*
* // *console.log(i);*
* // *i++;*
* // *}*
* // *let i = 10;*
* // *do{*
* // *console.log(i);*
* // *i++;*
* // *}while(i<=9);*
* // *console.log("value of i is ", i);*

# Arrays in JavaScript

* Intro to arrays
* // *intro to arrays*
* // *reference type*
* // *how to create arrays*
* // *ordered collection of items*
* // *let fruits = ["apple", "mango", "grapes"];*
* // *let numbers = [1,2,3,4];*
* // *let mixed = [1,2,2.3, "string", null, undefined];*
* // *console.log(mixed);*
* // *console.log(numbers);*
* // *console.log(fruits[2]);*
* letfruits=["apple","mango","grapes"];
* letobj={}; // *object literal*
* // *console.log(fruits);*
* // *fruits[1] = "banana";*
* // *console.log(fruits);*
* *console.log*(typeof fruits); //*object*
* *console.log*(typeof obj);
* *console.log*(*Array.isArray*(fruits));// *to check whether an array or not return true or false*
* *console.log*(*Array.isArray*(obj));
* // *array indexing*
* //*array are mutable*
* Push pop shift unshift
* // *array push pop*
* // *array shift unshift*
* letfruits=["apple","mango","grapes"];
* *console.log*(fruits);
* // *push insert element at last*
* // *fruits.push("banana");*
* // *console.log(fruits);*
* // *pop remove elements from last*
* // *let poppedFruit = fruits.pop();*
* // *console.log(fruits);*
* // *console.log("popped fruits is", poppedFruit);*
* // *unshift insert elm at start*
* // *fruits.unshift("banana");*
* // *fruits.unshift("myfruit");*
* // *console.log(fruits);*
* // *shift remove element from start*
* // *let removedFruit = fruits.shift();*
* // *console.log(fruits);*
* // *console.log("removed fruits is ", removedFruit);*
* Primitive vs reference data types
* // *primitve vs reference data types*
* //*primitve means only one vale (Stack)*
* //*reference data types contain address of variable (Heap)*
* // *let num1 = 6;*
* // *let num2 = num1;*
* // *console.log("value is num1 is", num1);*
* // *console.log("value is num2 is", num2);*
* // *num1++;*
* // *console.log("after incrementing num1")*
* // *console.log("value is num1 is", num1);*
* // *console.log("value is num2 is", num2);*
* // *reference types*
* // *array*
* letarray1=["item1","item2"];
* letarray2=array1;  //*here array2 will get the address of array1*
* *console.log*("array1", array1);
* *console.log*("array2", array2);
* *array1.push*("item3");
* *console.log*("after pushing element to array 1");
* *console.log*("array1", array1);
* *console.log*("array2", array2);
* Clone array & spread operator
* // *how to clone array*
* // *how to concatenate two arrays*
* letarray1=["item1","item2"];
* // *let array2 = ["item1", "item2"];*
* // *let array2 = array1.slice(0).concat(["item3", "item4"]);*
* // *let array2 = [].concat(array1,["item3", "item4"]); fastest way for cloning*
* // *new way*
* // *\*\*\*\*\*\*\*\*\*\*spread operator \*\*\*\*\*\*\*\*\*\*\*\*\**
* letoneMoreArray=["item3","item4"]
* letarray2=[...array1,...oneMoreArray];
* *array1.push*("item3");
* *console.log*(array1 === array2);
* *console.log*(array1)
* *console.log*(array2)
* For loop
* // *for loop in array*
* letfruits=["apple","mango","grapes","banana"];
* // *for(let i=0; i<=9;i++){*
* // *console.log(i);*
* // *}*
* // *console.log(fruits.length);*
* // *console.log(fruits[fruits.length-2]);*
* letfruits2=[];
* *for*(leti=0; i < *fruits.*length; i++){
* *fruits2.push*(fruits[i]*.toUpperCase*());
* }
* *console.log*(fruits2);
* use const for creating arrays
* // *use const for creating array*
* // *heap memory ["apple", "mango"] 0x11*
* // *const fruits = ["apple", "mango"]; // 0x11*
* // *fruits.push("banana");*
* // *console.log(fruits);*
* While loop in array
* // *use const for creating array*
* // *heap memory ["apple", "mango"] 0x11*
* // *const fruits = ["apple", "mango"]; // 0x11*
* // *fruits.push("banana");*
* // *console.log(fruits);*
* For of loop
* // *for of loop in array give the element direct*
* const *fruits* =["apple","mango","grapes","fruit4","fruit5"];
* const *fruits2* =[];
* // *for(let fruit of fruits){*
* // *fruits2.push(fruit.toUpperCase());*
* // *}*
* // *console.log(fruits2);*
* // *for(let i = 0; i<fruits.length; i++){*
* // *console.log(fruits[i]);*
* // *}*
* For in loop
* // *for in loop in array gives the index*
* const *fruits* =["apple","mango","grapes","fruit4","fruit5"];
* const *fruits2* =[];
* *for* (letindexin fruits) {
* *fruits2.push*(fruits[index]*.toUpperCase*());
* }
* *console.log*(fruits2);
* Array destructuring
* // *array destructuring*
* const *myArray* =["value1","value2","value3","value4"];
* // *let myvar1 = myArray[0];*
* // *let myvar2 = myArray[1];*
* // *console.log("value of myvar1", myvar1);*
* // *console.log("value of myvar2", myvar2);*
* let[myvar1,myvar2,...myNewArray]=myArray;
* *console.log*("value of myvar1", myvar1);
* *console.log*("value of myvar2", myvar2);
* *console.log*(myNewArray);

# Objects in JavaScript

* Intro to objects
* // *objects reference type*
* // *arrays are good but not sufficient*
* // *for real world data*
* // *objects store key value pairs*
* // *objects don't have index*
* // *how to create objects*
* // *const person = {name:"Harshit",age:22};*
* const *person* ={
* *name*:"harshit",
* *age*:22,
* //*"full name" : "anupam singh",*
* *hobbies*:["guitar","sleeping","listening music"]
* }
* *console.log*(person);
* // *how to access data from objects*
* // *console.log(person["name"]);*
* // *console.log(person["age"]);*
* // *console.log(person.hobbies);*
* // *how to add key value pair to objects*
* person["person"] = "male";
* // *person.person = "male";*
* *console.log*(person);
* Dot vs Bracket Notation
* // *difference between dot and bracket notaion*
* // *const key = "email";*
* // *const person = {*
* // *name: "harshit",*
* // *age: 22,*
* // *"person hobbies": ["guitar", "sleeping", "listening music"]*
* // *}*
* // *console.log(person["person hobbies"]);*
* // *person[key] = "harshitvashisth@gmail.com";*
* // *console.log(person);*
* //*in square brackets we can access value with space in between them*
* // *but with dot notation accces the value with space not possible*
* Iterate objects
* // *how to iterate object*
* const *person* ={
* *name*:"harshit",
* *age*:22,
* "person hobbies":["guitar","sleeping","listening music"]
* }
* // *for in loop*
* // *Object.keys returns array*
* // *for (let key in person) {*
* // *// console.log(`${key} : ${person[key]}`); //return a string*
* // *console.log(key, " : ", person[key]);*
* // *}*
* // *output:*
* // *name: harshit*
* // *age: 22*
* // *person hobbies: ['guitar', 'sleeping', 'listening music']*
* // *console.log(typeof (Object.keys(person))); //return an array of object*
* // *const val = Array.isArray((Object.keys(person))); array.isarray used to check weather or not*
* // *console.log(val);*
* // *for (let key of Object.keys(person)) {*
* // *console.log(person[key]);*
* // *}*
* // *output:*
* // *harshit*
* // *22*
* // *['guitar', 'sleeping', 'listening music']*
* Computed properties
* // *computed properties*
* const *key1* ="objkey1";
* const *key2* ="objkey2";
* const *value1* ="myvalue1";
* const *value2* ="myvalue2";
* // *const obj = {*
* // *objkey1 : "myvalue1",*
* // *objkey2 : "myvalue2",*
* // *}*
* // *const obj = {*
* // *[key1] : value1,*
* // *[key2] : value2*
* // *}*
* const *obj* ={};
* obj[key1] = value1;
* obj[key2] = value2;
* *console.log*(obj);
* Spread operator in objects
* // *spread operator*
* // *const array1 = [1, 2, 3];*
* // *const array2 = [5, 6, 7];*
* // *// const newArray = [...array1, ...array2, 89, 69];*
* // *const newArray = [..."123456789"];// separate object in an array*
* // *console.log(newArray);*
* // *spread operator in objects*
* const *obj1* ={
* *key1*:"value1",
* *key2*:"value2",
* //*key1: "value3" same value can't be possible*
* };
* const *obj2* ={
* *key1*:"valueUnique",
* *key3*:"value3",
* *key4*:"value4",
* };
* // *const newObject = { ...obj2, ...obj1, key69: "value69" };// key1 of obj2 will be added*
* // *const newObject = { ...["item1", "item2"] };*
* const *newObject* ={..."abcdefghijklmnopqrstuvwxyz"};
* *console.log*(newObject);
* // *{*
* // *'0': 'a',*
* // *'1': 'b',*
* // *'2': 'c',*
* // *'3': 'd',*
* // *'4': 'e',*
* // *'5': 'f',*
* // *'6': 'g',*
* // *'7': 'h',*
* // *'8': 'i',*
* // *'9': 'j',*
* // *'10': 'k',*
* // *'11': 'l',*
* // *'12': 'm',*
* // *'13': 'n',*
* // *'14': 'o',*
* // *'15': 'p',*
* // *'16': 'q',*
* // *'17': 'r',*
* // *'18': 's',*
* // *'19': 't',*
* // *'20': 'u',*
* // *'21': 'v',*
* // *'22': 'w',*
* // *'23': 'x',*
* // *'24': 'y',*
* // *'25': 'z'*
* // *}*
* Object Destructuring
* // *object destructuring*
* const *band* ={
* *bandName*:"led zepplin",
* *famousSong*:"stairway to heaven",
* *year*:1968,
* *anotherFamousSong*:"kashmir",
* };
* let{bandName,famousSong,...restProps}=band;
* *console.log*(bandName);
* *console.log*(restProps);
* Objects inside Array
* // *objects inside array*
* // *very useful in real world applications*
* const *users* =[
* {*userId*:1,*firstName*:'harshit', *gender*:'male'},
* {*userId*:2,*firstName*:'mohit', *gender*:'male'},
* {*userId*:3,*firstName*:'nitish', *gender*:'male'},
* ]
* *for*(letuserof users){
* *console.log*(*user.*firstName);
* }
* Nested Destructuring
* // *nested destructuring*
* const *users* =[
* {*userId*:1,*firstName*:'harshit', *gender*:'male'},
* {*userId*:2,*firstName*:'mohit', *gender*:'male'},
* {*userId*:3,*firstName*:'nitish', *gender*:'male'},
* ]
* const[{*firstName*: *user1firstName*, *userId*},,{*gender*: *user3gender*}]=users;
* *console.log*(user1firstName);
* *console.log*(userId);
* *console.log*(user3gender);

# Functions in JavaScript

* Function declaration
* function *singHappyBirthday*(){
* *console.log*("happy birthday to you ......");
* }
* function *sumThreeNumbers*(number1, number2, number3){
* *return* number1 + number2 + number3;
* }
* // *isEven*
* // *input : 1 number*
* // *output : true , false*
* // *function isEven(number){*
* // *return number % 2 === 0;*
* // *}*
* // *console.log(isEven(4));*
* // *function*
* // *input : string*
* // *output: firstCharacter*
* // *function firstChar(anyString){*
* // *return anyString[0];*
* // *}*
* // *console.log(firstChar("zbc"));*
* // *function*
* // *input : array, target (number)*
* // *output: index of target if target present in array*
* function *findTarget*(array, target){
* *for*(leti=0; i<*array.*length; i++){
* if(array[i]===target){
* *return* i;
* }
* }
* *return* -1;
* }
* const *myArray* =[1,3,8,90]
* const *ans* = *findTarget*(myArray,4);
* *console.log*(ans);
* Function Expression
* // *function expression*
* // *function singHappyBirthday(){*
* // *console.log("happy birthday to you ......");*
* // *}*
* const *singHappyBirthday* =function(){
* *console.log*("happy birthday to you ......");
* }
* // *singHappyBirthday();*
* const *sumThreeNumbers* =function(*number1*, *number2*, *number3*){
* *return* number1+number2+number3;
* }
* const *ans* = *sumThreeNumbers*(2,3,4);
* // *console.log(ans);*
* // *function isEven(number){*
* // *return number % 2 === 0;*
* // *}*
* const *isEven* =function(*number*){
* *return* number%2===0;
* }
* // *console.log(isEven(2));*
* const *firstChar* =function(*anyString*){
* *return* anyString[0];
* }
* const *findTarget* =function(*array*, *target*){
* *for*(leti=0;i<*array.length*;i++){
* if(array[i]===target){
* *return* i;
* }
* }
* *return* -1;
* }
* Arrow Functions
* // *arrow functions*
* // *const singHappyBirthday = function(){*
* // *console.log("happy birthday to you ......");*
* // *}*
* const *singHappyBirthday* =()=>{
* *console.log*("happy birthday to you ......");
* }
* *singHappyBirthday*();
* const *sumThreeNumbers* =(*number1*, *number2*, *number3*)=>{
* *return* number1+number2+number3;
* }
* const *ans* = *sumThreeNumbers*(2,3,4);
* *console.log*(ans);
* // *const isEven = function(number){*
* // *return number % 2 === 0;*
* // *}*
* const *isEven* = *number* =>number%2===0;
* *console.log*(*isEven*(4));
* const *firstChar* = *anyString* =>anyString[0];
* *console.log*(*firstChar*("harshit"));
* const *findTarget* =(*array*, *target*)=>{
* *for*(leti=0;i<*array.length*;i++){
* if(array[i]===target){
* *return* i;
* }
* }
* *return* -1;
* }
* Function declarations are hoisted (covered in great detail , later in this course)
* // *hoisting - declare a function before creation only possible with function declaration function hello() not with function expression var name= hello()*
* // *hello();*
* // *function hello(){*
* // *console.log("hello world");*
* // *}*
* // *console.log(hello);*
* // *const hello = "hello world";*
* // *console.log(hello);*
* Function inside function
* // *functions inside function*
* function *app*(){
* const *myFunc* =()=>{
* *console.log*("hello from myFunc")
* }
* const *addTwo* =(*num1*, *num2*)=>{
* *return* num1+num2;
* }
* const *mul* =(*num1*, *num2*)=>num1\*num2;
* *console.log*("inside app");
* *myFunc*();
* *console.log*(*addTwo*(2,3));
* *console.log*(*mul*(2,3));
* }
* *app*();
* Lexical Scope
* // *lexical scope*
* const *myVar* ="value1";
* function *myApp*(){
* function *myFunc*(){
* // *const myVar = "value59";*
* const *myFunc2* =()=>{
* *console.log*("inside myFunc",myVar);
* }
* *myFunc2*();
* }
* *console.log*(myVar);
* *myFunc*();
* }
* *myApp*();
* Block Scope Vs Function Scope
* // *block scope vs function scope*
* // *let and const are block scope*
* // *var is function scope*
* // *if(true){*
* // *var firstName = "harshit";*
* // *console.log(firstName);*
* // *}*
* // *console.log(firstName);*
* function *myApp*(){
* if(true){
* varfirstName="harshit";
* *console.log*(firstName);
* }
* if(true){
* *console.log*(firstName);
* }
* *console.log*(firstName);
* }
* *myApp*();
* Default Parameters
* // *default parameters*
* // *function addTwo(a,b){*
* // *if(typeof b ==="undefined"){*
* // *b = 0;*
* // *}*
* // *return a+b;*
* // *}*
* function *addTwo*(a,b=0){
* *return* a+b;
* }
* const *ans* = *addTwo*(4,8);
* *console.log*(ans);
* Rest Parameters
* // *rest parameters*
* // *function myFunc(a,b,...c){*
* // *console.log(`a is ${a}`);*
* // *console.log(`b is ${b}`);*
* // *console.log(`c is`, c);*
* // *}*
* // *myFunc(3,4,5,6,7,8,9);*
* function *addAll*(...numbers) {
* lettotal=0;
* *for* (letnumberof numbers) {
* total = total + number;
* }
* *return* total;
* }
* const *ans* = *addAll*(4,5,4,2,10);
* *console.log*(ans);
* Parameter Destructuring
* // *param destructuring*
* // *object*
* // *react*
* const *person* ={
* *firstName*:"harshit",
* *gender*:"male",
* *age*:500
* }
* // *function printDetails(obj){*
* // *console.log(obj.firstName);*
* // *console.log(obj.gender);*
* // *}*
* function *printDetails*({firstName, gender, age}){
* *console.log*(firstName);
* *console.log*(gender);
* *console.log*(age);
* }
* *printDetails*(person);
* Very brief intro to callback functions(covered in great detail , later in the course)
* // *callback functions*
* function *myFunc2*(name){
* *console.log*("inside my func 2")
* *console.log*(`your name is ${name}`);
* }
* function *myFunc*(callback){
* *console.log*("hello there I am a func and I can..")
* *callback*("harshit");
* }
* *myFunc*(myFunc2);
* Functions returning Functions
* // *function returning function*
* function *myFunc*(){
* function *hello*(){
* *return* "hello world"
* }
* *return* hello;
* }
* const *ans* = *myFunc*();
* *console.log*(*ans*());

# Very Important Array Methods

* Foreach method
* // *important array methods*
* const *numbers* =[4,2,5,8];
* // *function myFunc(number, index){*
* // *console.log(`index is ${index} number is ${number}`);*
* // *}*
* //*number,index,array*
* // *numbers.forEach(function(number,index){*
* // *console.log(`index is ${index} number is ${number}`);*
* // *});*
* // *numbers.forEach(function(number, index){*
* // *console.log(number\*3, index);*
* // *})*
* const *users* =[
* { *firstName*:"harshit", *age*:23},
* { *firstName*:"mohit", *age*:21},
* { *firstName*:"nitish", *age*:22},
* { *firstName*:"garima", *age*:20},
* ]
* // *users.forEach(function(user){*
* // *console.log(user.firstName);*
* // *});*
* // *users.forEach((user, index)=>{*
* // *console.log(user.firstName, index);*
* // *})*
* // *for(let user of users){*
* // *console.log(user.firstName);*
* // *}*
* Map method

// *map method -makes new array and store value*

// *const numbers = [3,4,6,1,8];*

// *const square = function(number){*

// *return number\*number; \*\*prefered*

// *console,log(number\*number);//not use when using map because it cause undefined value in array made by map method only return the value*

// *}*

//*cont squareNumber = numbers.map(square);*

//*[9,16,36,1,64]*

//*map method*

// *const squareNumber = numbers.map((number, index)=>{*

// *return index;*

// *});*

// *console.log(squareNumber);*

const *users* =[

{ *firstName*:"harshit", *age*:23},

{ *firstName*:"mohit", *age*:21},

{ *firstName*:"nitish", *age*:22},

{ *firstName*:"garima", *age*:20},

]

const *userNames* = *users.map*((*user*)=>{

*return user.firstName*;

});

*console.log*(userNames);

* Filter
* // *filter method*
* const *numbers* =[1,3,2,6,4,8];
* const *evenNumbers* = *numbers.filter*((*number*)=>{
* *return* number%2===0;
* });
* *console.log*(evenNumbers);
* Reduce
* // *reduce*
* const *numbers* =[1,2,3,4,5,10];
* // *aim : sum of all the numbers in array*
* // *const sum = numbers.reduce((accumulator, currentValue)=>{*
* // *return accumulator + currentValue;*
* // *}, 100);*
* //*100 is initial value of accumulator*
* // *console.log(sum);*
* // *accumulator , currentValue,  return*
* // *1               2              3*
* // *3               3              6*
* // *6               4              10*
* // *10              5              15*
* // *15              10             25*
* // *const userCart = [*
* // *{productId: 1, productName: "mobile", price: 12000},*
* // *{productId: 2, productName: "laptop", price: 22000},*
* // *{productId: 3, productName: "tv", price: 15000},*
* // *]*
* // *const totalAmount = userCart.reduce((totalPrice, currentProduct)=>{*
* // *return totalPrice + currentProduct.price;*
* // *}, 0)*
* // *console.log(totalAmount);*
* // *total price      currentValue     return*
* // *0                {}                  12000*
* // *12000            22000                34000*
* // *34000            15000                49000*
* Sort
* // *sort method*
* // *ASCII TABLE*
* //*char : ascii value*
* // *'0' : 48*
* // *'1' : 49*
* // *'2' : 50*
* // *'3' : 51*
* // *'4' : 52*
* // *'5' : 53*
* // *'6' : 54*
* // *'7' : 55*
* // *'8' : 56*
* // *'9' : 57*
* // *':' : 58*
* // *';' : 59*
* // *'<' : 60*
* // *'=' : 61*
* // *'>' : 62*
* // *'?' : 63*
* // *'@' : 64*
* // *'A' : 65*
* // *'B' : 66*
* // *'C' : 67*
* // *'D' : 68*
* // *'E' : 69*
* // *'F' : 70*
* // *'G' : 71*
* // *'H' : 72*
* // *'I' : 73*
* // *'J' : 74*
* // *'K' : 75*
* // *'L' : 76*
* // *'M' : 77*
* // *'N' : 78*
* // *'O' : 79*
* // *'P' : 80*
* // *'Q' : 81*
* // *'R' : 82*
* // *'S' : 83*
* // *'T' : 84*
* // *'U' : 85*
* // *'V' : 86*
* // *'W' : 87*
* // *'X' : 88*
* // *'Y' : 89*
* // *'Z' : 90*
* // *'[' : 91*
* // *'\' : 92*
* // *']' : 93*
* // *'^' : 94*
* // *'\_' : 95*
* // *'`' : 96*
* // *'a' : 97*
* // *'b' : 98*
* // *'c' : 99*
* // *'d' : 100*
* // *'e' : 101*
* // *'f' : 102*
* // *'g' : 103*
* // *'h' : 104*
* // *'i' : 105*
* // *'j' : 106*
* // *'k' : 107*
* // *'l' : 108*
* // *'m' : 109*
* // *'n' : 110*
* // *'o' : 111*
* // *'p' : 112*
* // *'q' : 113*
* // *'r' : 114*
* // *'s' : 115*
* // *'t' : 116*
* // *'u' : 117*
* // *'v' : 118*
* // *'w' : 119*
* // *'x' : 120*
* // *'y' : 121*
* // *'z' : 122*
* // *'{' : 123*
* // *'|' : 124*
* // *'}' : 125*
* // *sort*
* // *5,9,1200, 400, 3000*
* // *5, 9, 400, 1200, 3000 (expected)*
* // *["5", "9", "1210", "410", "3000"]*
* // *[53, 57, 49, 52, 51]*
* // *const userNames = ['harshit', 'abcd', 'mohit', 'nitish', 'aabc', 'ABC', 'Harshit'];*
* // *userNames.sort();*
* // *console.log(userNames);*
* // *const numbers = [5,9,1200, 410, 3000];*
* // *numbers.sort((a,b)=>{*
* // *return b-a; descending order m sortingkrega*
* // *});*
* // *numbers.sort((a,b)=>a-b); asscending order m sorting krega*
* // *console.log(numbers); return the sorted array on the basis of their value integer*
* // *1200,410*
* // *a-b ---> 790*
* // *a-b ---> postive (greater than 0) ---> b, a*
* // *410 , 1200*
* // *a-b ---> negative ----> a,b*
* // *5, 9 ---> -4*
* // *price lowToHigh HighToLow*
* const *products* =[
* { *productId*:1, *produceName*:"p1", *price*:300},
* { *productId*:2, *produceName*:"p2", *price*:3000},
* { *productId*:3, *produceName*:"p3", *price*:200},
* { *productId*:4, *produceName*:"p4", *price*:8000},
* { *productId*:5, *produceName*:"p5", *price*:500},
* ]
* // *lowToHigh*
* const *lowToHigh* = *products.slice*(0)*.sort*((*a*, *b*)=>{
* *return a.price* - *b.price*
* });
* const *highToLow* = *products.slice*(0)*.sort*((*a*, *b*)=>{
* *return b.price* - *a.price*;
* });
* const *users* =[
* { *firstName*:"harshit", *age*:23},
* { *firstName*:"mohit", *age*:21},
* { *firstName*:"nitish", *age*:22},
* { *firstName*:"garima", *age*:20},
* ]
* *users.sort*((a, b) => {
* if (*a.*firstName > *b.*firstName) {
* *return* 1;
* } else {
* *return* -1;
* }
* });
* *console.log*(users);
* Find
* // *find method*
* // *const myArray = ["Hello", "catt", "dog", "lion"];*
* // *function isLength3(string){*
* // *return string.length === 3;*
* // *}*
* // *const ans = myArray.find((string)=>string.length===3);*
* // *console.log(ans);*
* const *users* =[
* {*userId* :1, *userName*:"harshit"},
* {*userId* :2, *userName*:"harsh"},
* {*userId* :3, *userName*:"nitish"},
* {*userId* :4, *userName*:"mohit"},
* {*userId* :5, *userName*:"aaditya"},
* ];
* const *myUser* = *users.find*((*user*)=>*user.userId*===3);
* *console.log*(myUser);
* Every
* // *every method*
* // *const numbers = [2,4,6,9,10];*
* // *const ans = numbers.every((number)=>number%2===0);*
* // *console.log(ans);*
* const *userCart* =[
* {*productId*:1, *productName*:"mobile", *price*:12000},
* {*productId*:2, *productName*:"laptop", *price*:22000},
* {*productId*:3, *productName*:"tv", *price*:35000},
* ]
* const *ans* = *userCart.every*((*cartItem*)=>*cartItem.price* <30000);
* *console.log*(ans);
* Some
* // *some method*
* const *numbers* =[3,5,11,9];
* // *kya ek bhi number esa hai jo even hai*
* // *true*
* // *const ans = numbers.some((number)=>number%2===0);*
* // *console.log(ans);*
* const *userCart* =[
* {*productId*:1, *productName*:"mobile", *price*:12000},
* {*productId*:2, *productName*:"laptop", *price*:22000},
* {*productId*:3, *productName*:"tv", *price*:35000},
* {*productId*:3, *productName*:"macbook", *price*:25000},
* ]
* const *ans* = *userCart.some*((*cartItem*)=>*cartItem.price* >100000);
* *console.log*(ans);
* Fill method
* // *fill method*
* // *value , start , end*
* // *const myArray = new Array(10).fill(0);*
* // *console.log(myArray);*
* const *myArray* =[1,2,3,4,5,6,7,8];
* *myArray.fill*(0, 2, 5);
* *console.log*(myArray);
* //*[1,2,0,0,0,6,7,8]*
* Splice method
* // *splice method*
* // *start , delete , insert*
* const *myArray* =['item1','item2','item3'];
* // *delete*
* // *const deletedItem = myArray.splice(1, 2);*
* // *console.log("delted item", deletedItem);*
* // *insert*
* // *myArray.splice(1, 0,'inserted item');*
* // *insert and delete*
* const *deletedItem* = *myArray.splice*(1,2,"inserted item1","inserted item2")
* *console.log*("delted item", deletedItem);
* *console.log*(myArray);

# More useful things

* Iterables
* // *iterables*
* // *jispe hum for of loop laga sakein*
* // *string , array are iterable*
* // *const firstName = "Harshit";*
* // *for(let char of firstName){*
* // *console.log(char);*
* // *}*
* const *items* =['item1','item2','item3'];
* // *for(let item of items){*
* // *console.log(item);*
* // *}*
* // *array like object*
* // *jinke pas length property hoti hai*
* // *aur jiko hum index se access kar sakte hai*
* // *example :- string*
* // *const firstName = "harshit";*
* // *console.log(firstName.length);*
* // *console.log(firstName[2]);*
* Sets
* // *Sets (it is iterable)*
* // *store data*
* // *sets also have its own methods*
* // *No index-based access*
* // *Order is not guaranteed*
* // *unique items only (no duplicates allowed)*
* // *const items = ['item1', 'item2', 'item3'];*
* // *const numbers = new Set();*
* // *numbers.add(1);*
* // *numbers.add(2);*
* // *numbers.add(3);*
* // *numbers.add(4);*
* // *numbers.add(5);*
* // *numbers.add(6);*
* // *numbers.add(items);*
* // *if(numbers.has(1)){*
* // *console.log("1 is present")*
* // *}else{*
* // *console.log("1 is not present")*
* // *}*
* // *for(let number of numbers){*
* // *console.log(number);*
* // *}*
* // *const myArray = [1,2,4,4,5,6,5,6];*
* // *const uniqueElements = new Set(myArray);*
* // *let length = 0;*
* // *for(let element of uniqueElements){*
* // *length++;*
* // *}*
* // *console.log(length);*
* Maps
* // *Maps different from map method*
* // *map is an iterable*
* // *store data in ordered fashion*
* // *store key value pair (like object)*
* // *duplicate keys are not allowed like objects*
* // *different between maps and objects*
* // *objects can only have string or symbol*
* // *as key*
* // *in maps you can use anything as key*
* // *like array, number, string*
* // *object literal*
* // *key -> string*
* // *key -> symbol*
* // *const person = {*
* // *firstName : "harshit",*
* // *age: 7,*
* // *1:"one"*
* // *}*
* // *// console.log(person.firstName);*
* // *// console.log(person["firstName"]);*
* // *// console.log(person[1]);*
* // *for(let key in person){*
* // *console.log(typeof key);*
* // *}*
* // *key value pair*
* // *const person = new Map();*
* // *person.set('firstName', 'Harshit');*
* // *person.set('age', 7);*
* // *person.set(1,'one');*
* // *person.set([1,2,3],'onetwothree');*
* // *person.set({1: 'one'},'onetwothree');*
* // *console.log(person);*
* // *console.log(person.get(1));*
* // *console.log(person.keys()); mapiterator*
* // *for(let key of person.keys()){*
* // *console.log(key, typeof key);*
* // *}*
* // *for(let [key, value] of person){*
* // *// console.log(Array.isArray(key));*
* // *console.log(key, value)*
* // *}*
* const *person1* ={
* *id*:1,
* *firstName*:"harshit"
* }
* const *person2* ={
* *id*:2,
* *firstName*:"harshta"
* }
* const *extraInfo* =new *Map*();
* *extraInfo.set*(person1, { age: 8, gender: "male" });
* *extraInfo.set*(person2, { age: 9, gender: "female" });
* // *console.log(userInfo);*
* *console.log*(*person1.*id);
* *console.log*(*extraInfo.get*(person1)*.*gender);
* *console.log*(*extraInfo.get*(person2)*.*gender);
* Object.assign
* // *clone using Object.assign*
* // *memory*
* const *obj* ={
* *key1*:"value1",
* *key2*:"value2"
* }
* // *const obj2 = {'key69': "value69",...obj};*
* // *const obj2 = Object.assign({'key69': "value69"}, obj);*
* // *obj.key3 = "value3";*
* // *console.log(obj);*
* // *console.log(obj2);*
* Optional chaining
* // *optional chaining*
* const *user* ={
* *firstName*:"harshit",
* // *address: {houseNumber: '1234'}*
* }
* *console.log*(*user?.*firstName);
* *console.log*(*user?.address?.*houseNumber);
* // *console.log(user.address.houseNumber); undefined*

# Object Oriented JavaScript / Prototypal Inheritance

* Methods
* // *methods-function inside object*
* function *personInfo*() {
* *console.log*(`person name is ${this*.*firstName} and age is ${this*.*age}`);
* }
* const *person1* ={
* *firstName*:"harsh",
* *age*:8,
* *about*:personInfo
* }
* const *person2* ={
* *firstName*:"mohit",
* *age*:18,
* *about*:personInfo
* }
* const *person3* ={
* *firstName*:"nitish",
* *age*:17,
* *about*:personInfo
* }
* *person1.about*();
* *person2.about*();
* *person3.about*();
* This keyword, Window object
* // *console.log(window);*
* // *"use strict";*
* // *function myFunc(){*
* // *console.log(this);*
* // *}*
* // *myFunc();*
* //*return window object if strict mode not used*
* Outside of use strict mode it return undefined
* Call , apply and bind method
* function *about*(hobby, favMusician) {
* *console.log*(this*.*firstName, this*.*age, hobby, favMusician);
* }
* const *user1* ={
* *firstName*:"harshit",
* *age*:8,
* }
* const *user2* ={
* *firstName*:"mohit",
* *age*:9,
* }
* //call method help to call the function of differenct object and tells the object which value is assigned to this
* // *apply*
* // *about.apply(user1, ["guitar", "bach"]);*
* //apply or call method works same the difference is how we can pass the arguments
* // *about.call(user2,'guitar2','moazrt');*
* // *const func = about.bind(user2, "guitar", "bach");*
* // *func();*
* //bind function returns an function can’t use directly
* Some warnings
* const *user1* ={
* *firstName* :"harshit",
* *age*:8,
* *about*:function(){
* *console.log*(*this.firstName*, *this.age*);
* }
* }
* // *don't do this mistake*
* // *user1.about();*
* Const myfunc =user1.about(user1) will return undefinded on calling myfunc() that’s why bind
* const *myFunc* = *user1.about.bind*(user1);
* *myFunc*();
* This inside arrow functions
* // *arrow functions dont have this in their scope they find this in one level up*
* const *user1* ={
* *firstName*:"harshit",
* *age*:8,
* *about*:()=>{
* *console.log*(*this.firstName*, *this.age*);
* }
* }
* *user1.about*(user1);
* //*undefinded*
* Short syntax for methods
* // *const user1 = {*
* // *firstName : "harshit",*
* // *age: 8,*
* // *about: function(){*
* // *console.log(this.firstName, this.age);*
* // *}*
* // *}*
* // *const user1 = {*
* // *firstName : "harshit",*
* // *age: 8,*
* // *about(){*
* // *console.log(this.firstName, this.age);*
* // *}*
* // *}*
* *user1.about*();
* Factory functions & discuss some memory related problems
* // *function (that function create object)*
* // *2.) add key value pair*
* // *3.) object ko return krega*
* function *createUser*(firstName, lastName, email, age, address) {
* const *user* ={};
* *user.*firstName = firstName;
* *user.*lastName = lastName;
* *user.*email = email;
* *user.*age = age;
* *user.*address = address;
* *user.about* = function () {
* *return* `${this*.*firstName} is ${this*.*age} years old.`;
* };
* *user.is18* = function () {
* *return* this*.*age >= 18;
* }
* *return* user;
* }
* const *user1* = *createUser*('harshit','vashsith','harshit@gmail.com',19,"my address");
* *console.log*(user1);
* const *is18* = *user1.is18*();
* const *about* = *user1.about*();
* *console.log*(about);
* *//the problem in the solution is that everytime a new object is created with all its methods and properties which is not memory efficent*
* First solution to that problem
* const *userMethods* ={
* *about*:function(){
* *return* `${*this.firstName*} *is* ${*this.age*} *years old.*`;
* },
* *is18*:function(){
* *return this.age* >=18;
* }
* }
* function *createUser*(firstName, lastName, email, age, address) {
* const *user* ={};
* *user.*firstName = firstName;
* *user.*lastName = lastName;
* *user.*email = email;
* *user.*age = age;
* *user.*address = address;
* *user.*about = *userMethods.*about;
* *user.*is18 = *userMethods.*is18;
* *return* user;
* }
* /\*here all the methods separated which can be used anytime when needed helps to save memory\*\
* const *user1* = *createUser*('harshit','vashsith','harshit@gmail.com',9,"my address");
* const *user2* = *createUser*('harsh','vashsith','harshit@gmail.com',19,"my address");
* const *user3* = *createUser*('mohit','vashsitha','harshit@gmail.com',17,"my address");
* *console.log*(*user1.about*());
* *console.log*(*user3.about*());
* Why that solution isn’t that great
* const *userMethods* ={
* *about* :function(){
* *return* `${*this.firstName*} *is* ${*this.age*} *years old.*`;
* },
* *is18* :function(){
* *return this.age* >=18;
* },
* *sing*:function(){
* *return* 'toon na na na la la ';
* }
* }
* function *createUser*(firstName, lastName, email, age, address){
* const *user* = *Object.create*(userMethods);// *{}*
* *user.*firstName = firstName;
* *user.*lastName = lastName;
* *user.*email = email;
* *user.*age = age;
* *user.*address = address;
* *return* user;
* }
* const *user1* = *createUser*('harshit','vashsith','harshit@gmail.com',9,"my address");
* const *user2* = *createUser*('harsh','vashsith','harshit@gmail.com',19,"my address");
* const *user3* = *createUser*('mohit','vashsitha','harshit@gmail.com',17,"my address");
* *console.log*(user1);
* *console.log*(*user1.about*());
* // *console.log(user3.sing());*
* In this the all the methods passed as a prototype present in the function can be called anytime when required
* What is proto , [[prototype]]
* const *obj1* ={
* *key1*:"value1",
* *key2*:"value2"
* }
* // *\_\_proto\_\_*
* // *offical ecmascript documentation*
* // *[[prototype]]*
* // *\_\_proto\_\_ , [[prototype]]*
* // *prototype*
* const *obj2* = *Object.create*(obj1); // *{}*
* // *there is one more way to create empty object*
* New object with all the key value pair or methods present in it are made bydefault
* *obj2.*key3 = "value3";
* // *obj2.key2 = "unique";*
* *console.log*(obj2);
* *console.log*(*obj2.*\_\_proto\_\_);
* //*proto and prototype is different*
* What is prototype
* //sFunction provide some free space to add some function or values which can be used anytime
* function *hello*(){
* *console.log*("hello world");
* }
* // *javascript function ===> function  + object*
* // *console.log(hello.name);*
* // *you can add your own properties*
* // *hello.myOwnProperty = "very unique value";*
* // *console.log(hello.myOwnProperty);*
* // *name property ---> tells function name;*
* // *function provides more usefull properties.*
* // *console.log(hello.prototype); // {}*
* // *only functions provide prototype property*
* // *hello.prototype.abc = "abc";*
* // *hello.prototype.xyz = "xyz";*
* // *hello.prototype.sing = function(){*
* // *return "lalalla";*
* // *};*
* // *console.log(hello.prototype.sing());*
* Use prototype
* // *const userMethods = {*
* // *about : function(){*
* // *return `${this.firstName} is ${this.age} years old.`;*
* // *},*
* // *is18 : function(){*
* // *return this.age >= 18;*
* // *},*
* // *sing: function(){*
* // *return 'toon na na na la la ';*
* // *}*
* // *}*
* function *createUser*(firstName, lastName, email, age, address){
* const *user* = *Object.create*(*createUser.prototype*);// *{}*
* *user.*firstName = firstName;
* *user.*lastName = lastName;
* *user.*email = email;
* *user.*age = age;
* *user.*address = address;
* *return* user;
* }
* createUser*.*prototype*.about* = function(){
* *return* `${this*.*firstName} is ${this*.*age} years old.`;
* };
* createUser*.*prototype*.is18* = function (){
* *return* this*.*age >= 18;
* }
* createUser*.*prototype*.sing* = function (){
* *return* "la la la la ";
* }
* const *user1* = *createUser*('harshit','vashsith','harshit@gmail.com',18,"my address");
* const *user2* = *createUser*('harsh','vashsith','harshit@gmail.com',19,"my address");
* const *user3* = *createUser*('mohit','vashsitha','harshit@gmail.com',17,"my address");
* *console.log*(user1);
* *console.log*(*user1.is18*());
* New keyword
* // *new keyword*
* // *1.) this = {}*
* // *2.) return {}*
* //
* // *\_\_proto\_\_*
* // *// official ecmascript document*
* // *[[prototype]]*
* // *constructor function*
* function *CreateUser*(firstName, lastName, email, age, address){
* this*.*firstName = firstName;
* this*.*lastName = lastName;
* this*.*email = email;
* this*.*age = age;
* this*.*address = address;
* }
* CreateUser*.*prototype*.about* = function(){
* *return* `${this*.*firstName} is ${this*.*age} years old.`;
* };
* CreateUser*.*prototype*.is18* = function (){
* *return* this*.*age >= 18;
* }
* CreateUser*.*prototype*.sing* = function (){
* *return* "la la la la ";
* }
* const *user1* =new *CreateUser*('harshit','vashsith','harshit@gmail.com',18,"my address");
* const *user2* =new *CreateUser*('harsh','vashsith','harshit@gmail.com',19,"my address");
* const *user3* =new *CreateUser*('mohit','vashsitha','harshit@gmail.com',17,"my address");
* *console.log*(user1);
* *console.log*(*user1.is18*());
* Constructor function with new keyword
* function *CreateUser*(firstName, lastName, email, age, address){
* this*.*firstName = firstName;
* this*.*lastName = lastName;
* this*.*email = email;
* this*.*age = age;
* this*.*address = address;
* }
* CreateUser*.*prototype*.about* = function(){
* *return* `${this*.*firstName} is ${this*.*age} years old.`;
* };
* CreateUser*.*prototype*.is18* = function (){
* *return* this*.*age >= 18;
* }
* CreateUser*.*prototype*.sing* = function (){
* *return* "la la la la ";
* }
* const *user1* =new *CreateUser*('harshit','vashsith','harshit@gmail.com',18,"my address");
* const *user2* =new *CreateUser*('harsh','vashsith','harshit@gmail.com',19,"my address");
* const *user3* =new *CreateUser*('mohit','vashsitha','harshit@gmail.com',17,"my address");
* *for*(letkeyin user1){
* // *console.log(key);*
* if(*user1.hasOwnProperty*(key)){
* *console.log*(key);
* }
* }
* More discussion about proto and prototype
* // *let numbers = [1,2,3];*
* // *// console.log(Object.getPrototypeOf(numbers));*
* // *console.log(Array.prototype);*
* // *console.log(numbers);*
* // *function hello(){*
* // *console.log("hello");*
* // *}*
* Class keyword

// *2015 / es6*

// *class keyword*

// *class are fake*

class CreateUser{

    constructor(firstName, lastName, email, age, address){

        this*.*firstName = firstName;

        this*.*lastName = lastName;

        this*.*email = email;

        this*.*age = age;

        this*.*address = address;

    }

*about*(){

*return* `${this*.*firstName} is ${this*.*age} years old.`;

    }

*is18*(){

*return* this*.*age >= 18;

    }

*sing*(){

*return* "la la la la ";

    }

}

const *user1* =new *CreateUser*('harshit','vashsith','harshit@gmail.com',18,"my address");

const *user2* =new *CreateUser*('harsh','vashsith','harshit@gmail.com',19,"my address");

const *user3* =new *CreateUser*('mohit','vashsitha','harshit@gmail.com',17,"my address");

// *console.log(Object.getPrototypeOf(user1));*

* Example using class keyword
* class Animal {
* constructor(name, age){
* this*.*name = name;
* this*.*age = age;
* }
* *eat*(){
* *return* `${this*.*name} is eating`;
* }
* *isSuperCute*(){
* *return* this*.*age <= 1;
* }
* *isCute*(){
* *return* true;
* }
* }
* class Dog *extends* Animal{
* }
* const *tommy* =new *Dog*("tommy",3);
* *console.log*(tommy);
* *console.log*(*tommy.isCute*());
* Super keyword
* // *super*
* class Animal {
* constructor(name, age){
* this*.*name = name;
* this*.*age = age;
* }
* *eat*(){
* *return* `${this*.*name} is eating`;
* }
* *isSuperCute*(){
* *return* this*.*age <= 1;
* }
* *isCute*(){
* *return* true;
* }
* }
* class Dog *extends* Animal{
* constructor(name, age, speed){
* super(name,age);
* this*.*speed = speed;
* }
* *run*(){
* *return* `${this*.*name} is running at ${this*.*speed}kmph`
* }
* }
* // *object / instance*
* const *tommy* =new *Dog*("tommy",3,45);
* *console.log*(*tommy.run*());
* Method overriding
* // *same method in subclass*
* class Animal {
* constructor(name, age){
* this*.*name = name;
* this*.*age = age;
* }
* *eat*(){
* *return* `${this*.*name} is eating`;
* }
* *isSuperCute*(){
* *return* this*.*age <= 1;
* }
* *isCute*(){
* *return* true;
* }
* }
* class Dog *extends* Animal{
* constructor(name, age, speed){
* super(name,age);
* this*.*speed = speed;
* }
* *eat*(){
* *return* `Modified Eat : ${this*.*name} is eating`
* }
* *run*(){
* *return* `${this*.*name} is running at ${this*.*speed}kmph`
* }
* }
* // *object / instance*
* // *const tommy = new Dog("tommy", 3,45);*
* // *console.log(tommy.run());*
* // *console.log(tommy.eat());*
* const *animal1* =new *Animal*('sheru',2);
* *console.log*(*animal1.eat*());
* Getters and setters
* // *getter and setters*
* class Person{
* constructor(firstName, lastName, age){
* this*.*firstName = firstName;
* this*.*lastName = lastName;
* this*.*age = age;
* }
* *get* *fullName*(){
* *return* `${this*.*firstName} ${this*.*lastName}`
* }
* *set* *fullName*(fullName){
* *const* [*firstName*, *lastName*]= *fullName.split*(" ");
* this*.*firstName = firstName;
* this*.*lastName = lastName;
* }
* }
* const *person1* =new *Person*("harshit","sharma",5);
* // *console.log(person1.fullName());*
* // *console.log(person1.fullName);*
* // *person1.fullName = "mohit vashistha";*
* // *console.log(person1);*
* Static methods and properties
* // *static methods and properties*
* class Person{
* constructor(firstName, lastName, age){
* this*.*firstName = firstName;
* this*.*lastName = lastName;
* this*.*age = age;
* }
* *static* *classInfo*(){
* *return* 'this is person class';
* }
* *static* desc = "static property";
* *get* *fullName*(){
* *return* `${this*.*firstName} ${this*.*lastName}`
* }
* *set* *fullName*(fullName){
* *const* [*firstName*, *lastName*]= *fullName.split*(" ");
* this*.*firstName = firstName;
* this*.*lastName = lastName;
* }
* *eat*(){
* *return* `${this*.*firstName} is eating`;
* }
* *isSuperCute*(){
* *return* this*.*age <= 1;
* }
* *isCute*(){
* *return* true;
* }
* }
* const *person1* =new *Person*("harshit","sharma",8);
* // *// console.log(person1.eat());*
* // *const info = Person.classInfo();*
* // *console.log(person1.desc);*
* // *console.log(info);*

# How JavaScript Works

* Global Execution context
* // *compilation*
* // *code execution*
* // *why compilation*
* // *How javascript code executes*
* // *what is global exection context ?*
* // *what is local execution context ?*
* // *closures*
* *console.log*(this);
* *console.log*(window);
* *console.log*(firstName);
* varfirstName="Harshit";
* *console.log*(firstName);
* This and window in global execution context
* Hoisting
* // *hoisting*
* *console.log*(this);
* *console.log*(window);
* *console.log*(myFunction);
* *console.log*(fullName);
* function *myFunction*(){
* *console.log*("this is my function");
* }
* varfirstName="Harshit";
* varlastName="Sharma"
* varfullName=firstName+" "+lastName;
* *console.log*(fullName);
* Are let and const are hoisted ? What is a reference Error ?
* *console.log*(myFunction);
* var *myFunction* =function(){
* *console.log*("this is my function");
* }
* *console.log*(myFunction);
* // *Uncaught ReferenceError:*
* // *Cannot access 'firstName' before initialization*
* // *Uncaught ReferenceError:*
* // *firstName is not defined*
* // *console.log(firstName);*
* // *console.log(firstName);*
* // *let firstName;*
* // *console.log(firstName);*
* // *console.log(typeof firstName);*
* // *let firstName = "harshit";*
* *console.log*("hello world");
* letfirstName="Harshit";
* letlastName="Vashistha";
* const *myFunction* =function(){
* letvar1="First Variable";
* letvar2="second Variable";
* *console.log*(var1);
* *console.log*(var2);
* }
* Function execution context
* // *function execution context*
* letfoo="foo";
* *console.log*(foo);
* function *getFullName*(firstName, lastName){
* *console.log*(arguments);
* letmyVar="var inside func";
* *console.log*(myVar);
* const *fullName* =firstName+" "+lastName;
* *return* fullName;
* }
* const *personName* = *getFullName*("harshit","sharma");
* *console.log*(personName);
* Scope chain and lexical environment
* // *lexical environment, scope chain*
* const *lastName* ="Vashistha";
* const *printName* =function(){
* const *firstName* ="harshit";
* function *myFunction*(){
* *console.log*(firstName);
* *console.log*(lastName);
* }
* *myFunction*()
* }
* *printName*();
* Intro to closures
* // *closures*
* // *closure : 30-40%*
* // *analyse : 70-80%*
* // *real example : 100%*
* // *function can return functions*
* // *function outerFunction(){*
* // *function innerFunction(){*
* // *console.log("hello world")*
* // *}*
* // *return innerFunction;*
* // *}*
* // *const ans = outerFunction();*
* // *// console.log(ans);*
* // *ans();*
* function *printFullName*(firstName, lastName){
* function *printName*(){
* *console.log*(firstName, lastName);
* }
* *return* printName;
* }
* const *ans* = *printFullName*("harshit","sharma");
* // *console.log(ans);*
* *ans*();
* Closure example 1
* function *hello*(x){
* const *a*="varA";
* const *b* ="varB";
* *return* function(){
* *console.log*(a,b,x);
* }
* }
* const *ans* = *hello*("arg");
* *ans*();
* Closure Example 2
* // *function myFunction(power){*
* // *return function(number){*
* // *return number \*\* power*
* // *}*
* // *}*
* // *const square = myFunction(2);*
* // *const ans = square(3);*
* // *console.log(ans);*
* // *const cube = myFunction(3);*
* // *const ans2 = cube(3);*
* // *console.log(ans2);*
* function *myFunction*(power){
* *return* function(number){
* *return* number \*\* power
* }
* }
* const *square* = *myFunction*(2);
* const *ans* = *square*(3);
* *console.log*(ans);
* const *cube* = *myFunction*(3);
* const *ans2* = *cube*(3);
* *console.log*(ans2);
* Closure Example 3
* function *func*(){
* letcounter=0;
* *return* function(){
* if(counter < 1){
* *console.log*("Hi You Called me");
* counter++;
* }else{
* *console.log*("Mai already ek bar call ho chuka hoon!");
* }
* }
* }
* const *myFunc* = *func*();
* *myFunc*();
* *myFunc*();

# DOM Tutorial

* HTML and CSS Crash course ( Around 30-40 minutes)
* Async vs defer
* // *DOM*
* // *document object model*
* // *overview*
* // *how to use*
* // *deep study*
* // *console.dir(document);*
* Select elements using id
* // *select element using get element by id*
* const *mainHeading* = *document.getElementById*("main-heading");
* *console.log*(mainHeading);
* querySelector
* // *select element using query selector*
* // *const mainHeading = document.getElementById("main-heading");*
* const *mainHeading* = *document.querySelector*("#main-heading");
* const *header* = *document.querySelector*(".header");
* const *navItem* = *document.querySelectorAll*(".nav-item")
* *console.log*(navItem);
* textContent & innerText
* // *change text*
* // *textContent and innerText*
* // *const mainHeading = document.getElementById("main-heading");*
* //*.innerText can show all the text which shown on screen*
* //*.textContent can provide all the data in the tag visible or not*
* // *console.log(mainHeading.innerText);*
* // *mainHeading.textContent = "This is something else";*
* // *console.log(mainHeading.textContent);*
* Change the styles of elements using js
* // *change the styles of elements*
* const *mainHeading* = *document.querySelector*("div.headline h2");
* *console.log*(*mainHeading.*style);
* *mainHeading.style.*backgroundColor = "blue";
* *mainHeading.style.*border = "20px solid green";
* Get and set attributes
* // *get and set attrubutes*
* const *link* = *document.querySelector*("a");
* *console.log*(*link.getAttribute*("href")*.slice*(1));
* // *link.setAttribute("href", "https://codprog.com");*
* // *console.log(link.getAttribute("href"));*
* // *const inputElement = document.querySelector(".form-todo input");*
* // *console.log(inputElement.getAttribute("type"));*
* Select multiple elements and loop through them
* // *get multiple elements using getElements by class name*
* // *get multiple elements items using querySelectorAll*
* // *const navItems = document.getElementsByClassName("nav-item"); // HTMLCollection*
* // *console.log(navItems);*
* // *console.log(Array.isArray(navItems));*
* // *const navItems = document.querySelectorAll(".nav-item"); // NodeList*
* // *console.log(navItems[1]);*
* innerHTML
* // *get multiple elements using getElements by class name*
* // *get multiple elements items using querySelectorAll*
* // *array like object ---> indexing, length property*
* // *let navItems = document.getElementsByTagName("a"); // HTMLCollection*
* // *console.log(navItems);*
* // *we can't use forEach method to iterate through HTMLCollection*
* // *simple for loop*
* // *for of loop*
* // *forEach*
* // *for(let i=0; i< navItems.length; i++){*
* // *// console.log(navItems[i]);*
* // *const navItem = navItems[i];*
* // *navItem.style.backgroundColor = "#fff";*
* // *navItem.style.color = "green";*
* // *navItem.style.fontWeight = "bold";*
* // *}*
* // *for(let navItem of navItems){*
* // *navItem.style.backgroundColor = "#fff";*
* // *navItem.style.color = "green";*
* // *navItem.style.fontWeight = "bold";*
* // *}*
* // *navItems = Array.from(navItems);*
* // *console.log(Array.isArray(navItems));*
* // *navItems.forEach((navItem)=>{*
* // *navItem.style.backgroundColor = "#fff";*
* // *navItem.style.color = "green";*
* // *navItem.style.fontWeight = "bold";*
* // *})*
* // *console.log(Array.isArray(navItems));*
* // *const navItems = document.querySelectorAll(".nav-item"); // NodeList*
* // *console.log(navItems[1]);*
* // *let navItems = document.querySelectorAll("a");*
* // *navItems = Array.from(navItems);*
* // *console.log(Array.isArray(navItems));*
* // *simple for loop*
* // *for of loop*
* // *forEach*
* // *for(let i=0; i< navItems.length; i++){*
* // *// console.log(navItems[i]);*
* // *const navItem = navItems[i];*
* // *navItem.style.backgroundColor = "#fff";*
* // *navItem.style.color = "green";*
* // *navItem.style.fontWeight = "bold";*
* // *}*
* // *for(let navItem of navItems){*
* // *navItem.style.backgroundColor = "#fff";*
* // *navItem.style.color = "green";*
* // *navItem.style.fontWeight = "bold";*
* // *}*
* // *navItems.forEach((navItem)=>{*
* // *navItem.style.backgroundColor = "#fff";*
* // *navItem.style.color = "green";*
* // *navItem.style.fontWeight = "bold";*
* // *})*
* // *console.log(navItems);*

// *innerHtML*

const *headline* = *document.querySelector*(".headline");

// *console.log(headline.innerHTML);*

// *headline.innerHTML = "<h1>Inner html changed </h1>";*

// *headline.innerHTML += "<button class= \"btn\"> Learn More </button>"*

// *console.log(headline.innerHTML);*

* Deeply understand dom tree, root node , element nodes, text nodes
* // *const rootNode = document.getRootNode();*
* // *const htmlElementNode = rootNode.childNodes[0];*
* // *// console.log(htmlElementNode.childNodes); NodeList(3) [head, text, body]*
* // *const headElementNode = htmlElementNode.childNodes[0];*
* // *const textNode1 = htmlElementNode.childNodes[1];*
* // *const bodyElementNode = htmlElementNode.childNodes[2];*
* // *console.log(headElementNode.childNodes);*
* // *sibling relation*
* // *const h1 = document.querySelector("h1");*
* // *const body = h1.parentNode.parentNode;*
* // *body.style.color = "#efefef";*
* // *body.style.backgroundColor = "#333"*
* // *const body = document.body*
* // *body.style.color = "#efefef";*
* // *body.style.backgroundColor = "#333"*
* // *const head = document.querySelector("head");*
* // *// console.log(head);*
* // *const title = head.querySelector("title");*
* // *console.log(title.childNodes);*
* const *container* = *document.querySelector*(".container");
* *console.log*(*container.*children);
* classList
* // *const sectionTodo = document.querySelector(".section-todo");*
* // *console.log(sectionTodo.classList);*
* // *sectionTodo.classList.add('bg-dark');*
* // *sectionTodo.classList.remove("container");*
* // *const ans = sectionTodo.classList.contains("container");*
* // *console.log(ans);*
* // *sectionTodo.classList.toggle("bg-dark");*
* // *sectionTodo.classList.toggle("bg-dark");*
* const *header* = *document.querySelector*(".header");
* // *header.classList.add("bg-dark");*
* *console.log*(*header.*classList);
* Add new elements to page
* // *Add new HTML elements to page*
* // *innerHTML to add html element*
* const *todoList* = *document.querySelector*(".todo-list");
* // *console.log(todoList.innerHTML)*
* // *todoList.innerHTML = "<li>New Todo 2 </li>"*
* // *todoList.innerHTML += "<li>New Todo </li>";*
* // *todoList.innerHTML += "<li>teach students </li>";*
* // *when you should use it , when you should not*
* *todoList.insertAdjacentElement*("afterbegin", '<li>Hi</li>')
* Create elements
* // *document.createElement()*
* // *append*
* // *prepend*
* // *remove*
* // *const newTodoItem = document.createElement("li");*
* // *// const newTodoItemText = document.createTextNode("Teach students");*
* // *newTodoItem.textContent = "Teach students";*
* // *const todoList = document.querySelector(".todo-list");*
* // *todoList.prepend(newTodoItem);*
* // *console.log(newTodoItem);*
* // *const todo1 = document.querySelector('.todo-list li');*
* // *todo1.remove();*
* // *console.log(todo1)*
* // *before*
* // *after*
* // *const newTodoItem = document.createElement("li");*
* // *newTodoItem.textContent = "Teach students";*
* // *const todoList = document.querySelector(".todo-list");*
* // *todoList.after(newTodoItem);*
* Insert adjacent elements
* // *elem.insertAdjacentHTML(where, html)*
* // *beforebegin*
* // *afterbegin;*
* // *beforeend;*
* // *afterend;*
* // *const todoList = document.querySelector(".todo-list");*
* // *todoList.insertAdjacentHTML("beforeend", "<li>Teach Students </li>");*
* Clone nodes
* // *clone nodes*
* // *const ul = document.querySelector(".todo-list");*
* // *const li = document.createElement("li");*
* // *li.textContent = "new todo";*
* // *const li2 = li.cloneNode(true);*
* // *ul.append(li);*
* // *ul.prepend(li2);*
* More methods to add elements on page
* // *some old methods to support poor IE*
* // *appendChild;*
* // *insertBefore;*
* // *replaceChild;*
* // *removeChild*
* // *const ul = document.querySelector(".todo-list");*
* // *new element*
* // *const li = document.createElement("li");*
* // *li.textContent = "new todo";*
* // *const referenceNode = document.querySelector(".first-todo");*
* // *ul.removeChild(referenceNode);*

const *ul* = *document.querySelector*(".todo-list");

const *listItems* = *ul.getElementsByTagName*("li");

const *sixthLi* = *document.createElement*("li");

*sixthLi.*textContent = "item 6";

*ul.append*(sixthLi);

*console.log*(listItems);

* How to get the dimensions of the element

// *how to get the dimension of element*

// *height width*

const *sectionTodo* = *document.querySelector*(".section-todo");

const *info* = *sectionTodo.getBoundingClientRect*();

*console.log*(info);

* Intro to events
* // *intro to events*
* // *click*
* // *event add karne ke 3 tarike hai*
* // *1.)*
* const *btn* = *document.querySelector*(".btn-headline");
* // *method --- addEventListener*
* // *function clickMe(){*
* // *console.log("you clicked me !!!!!");*
* // *}*
* // *btn.addEventListener("click", function(){*
* // *console.log("you clicked me !!!!");*
* // *});*
* // *btn.addEventListener("click", ()=>{*
* // *console.log("arrow function !!!")*
* // *});*
* This keyword inside eventListener callback
* // *this keyword*
* const *btn* = *document.querySelector*(".btn-headline");
* *btn.addEventListener*("click",function(){
* *console.log*("you clicked me !!!!");
* *console.log*("value of this")
* *console.log*(this);
* });
* Add events on multiple elements
* const *allButtons* = *document.querySelectorAll*(".my-buttons button");
* // *for(let button of allButtons){*
* // *button.addEventListener("click", function(){*
* // *console.log(this);*
* // *})*
* // *}*
* // *for(let i = 0 ; i< allButtons.length; i++){*
* // *allButtons[i].addEventListener("click", function(){*
* // *console.log(this);*
* // *})*
* // *}*
* // *allButtons.forEach(function(button){*
* // *button.addEventListener("click", function(){*
* // *console.log(this);*
* // *});*
* // *})*
* Event object
* // *event object*
* // *const firstButton = document.querySelector("#one");*
* // *firstButton.addEventListener("click", function(event){*
* // *console.log(event);*
* // *})*
* // *jab bhi mai kisi bhi element pe event listener add hoga*
* // *js Engine --- line by line execute karta hai*
* // *browser ---- js Engine + extra features*
* // *browser ----- js Engine + WebApi*
* // *jab browser ko pata chala ki user ne event perform kia*
* // *jo hum listen kar rahe hai*
* // *browser ----- 2*
* // *1.) callback function hai vo js Engine ko degi ......*
* // *2.)  callback function ke sath browser jo event hua hai uski information bhi dega*
* // *ye info hamein ek object ke form mai milegi*
* const *allButtons* = *document.querySelectorAll*(".my-buttons button");
* *for*(letbuttonof allButtons){
* *button.addEventListener*("click",(e)=>{
* *console.log*(*e.*currentTarget);
* })
* }
* How event listener works
* *console.log*("script start !!!!!")
* const *allButtons* = *document.querySelectorAll*(".my-buttons button");
* *allButtons.forEach*((button)=>{
* *button.addEventListener*("click", (e)=>{
* letnum=0;
* *for*(leti=0; i<= 1000000000; i++){
* num += i;
* }
* *console.log*(*e.currentTarget.*textContent, num);
* })
* })
* letouterVar=0;
* *for*(leti=0; i<= 100000000; i++){
* outerVar += i;
* }
* *console.log*("value of outer variable is ", outerVar);
* *console.log*("script end !!!!!")
* Practice with events
* // *little practice with click event*
* const *allButtons* = *document.querySelectorAll*(".my-buttons button")
* // *console.log(allButtons.length);*
* *allButtons.forEach*(button =>{
* *button.addEventListener*("click", (e)=>{
* // *console.log(e.target);*
* *e.target.style.*backgroundColor = "yellow";
* *e.target.style.*color = "#333";
* })
* })
* Create demo project
* const *mainButton* = *document.querySelector*("button");
* const *body* = *document.body*;
* const *currentColor* = *document.querySelector*(".current-color");
* function *randomColorGenerator*(){
* const *red* = *Math.floor*(*Math.random*()\*256);
* const *green* = *Math.floor*(*Math.random*()\*256);
* const *blue* = *Math.floor*(*Math.random*()\*256);
* const *randomColor* =`*rgb(*${red}*,* ${green}*,* ${blue}*)*`
* *return* randomColor;
* }
* *mainButton.addEventListener*("click",()=>{
* const *randomColor* = *randomColorGenerator*();
* *body.style.*backgroundColor = randomColor;
* *currentColor.*textContent = randomColor;
* })
* More events
* // *keypress event*
* // *mouseover event*
* // *const body = document.body;*
* // *body.addEventListener("keypress", (e) => {*
* // *console.log(e.key);*
* // *});*
* // *const mainButton = document.querySelector(".btn-headline");*
* // *console.log(mainButton);*
* // *mainButton.addEventListener("mouseover", () => {*
* // *console.log("mouseover event ocurred!!!");*
* // *});*
* // *mainButton.addEventListener("mouseleave", () => {*
* // *console.log("mouseleave event ocurred!!!");*
* // *});*
* Event bubbling
* // *console.log("hello world");*
* const *grandparent* = *document.querySelector*(".grandparent");
* // *const parent = document.querySelector(".parent");*
* // *const child = document.querySelector(".child");*
* // *capturing events*
* // *child.addEventListener(*
* // *"click",*
* // *() => {*
* // *console.log("capture !!!! child");*
* // *},*
* // *true*
* // *);*
* // *parent.addEventListener(*
* // *"click",*
* // *() => {*
* // *console.log("capture !!!! parent");*
* // *},*
* // *true*
* // *);*
* // *grandparent.addEventListener(*
* // *"click",*
* // *() => {*
* // *console.log("capture !!!! grandparent");*
* // *},*
* // *true*
* // *);*
* // *document.body.addEventListener(*
* // *"click",*
* // *() => {*
* // *console.log("capture !!!! document.body");*
* // *},*
* // *true*
* // *);*
* //*/// not capture*
* // *child.addEventListener("click", () => {*
* // *console.log("bubble child");*
* // *});*
* // *parent.addEventListener("click", () => {*
* // *console.log("bubble parent");*
* // *});*
* // *grandparent.addEventListener("click", () => {*
* // *console.log("bubble grandparent");*
* // *});*
* // *document.body.addEventListener("click", () => {*
* // *console.log("bubble document.body");*
* // *});*
* // *event delegation*
* // *grandparent.addEventListener("click", (e) => {*
* // *console.log(e.target);*
* // *});*
* Event Capturing
* Event delegation
* Create Project using event delegation
* const *todoForm* = *document.querySelector*(".form-todo");
* const *todoInput* = *document.querySelector*(".form-todo input[type='text']");
* const *todoList* = *document.querySelector*(".todo-list");
* *todoForm.addEventListener*("submit", (e) => {
* *e.preventDefault*();
* const *newTodoText* = *todoInput.value*;
* const *newLi* = *document.createElement*("li");
* const *newLiInnerHtml* =`
* *<span class="text">*${newTodoText}*</span>*
* *<div class="todo-buttons">*
* *<button class="todo-btn done">Done</button>*
* *<button class="todo-btn remove">Remove</button>*
* *</div>*`;
* *newLi.*innerHTML = newLiInnerHtml;
* *todoList.append*(newLi);
* *todoInput.*value = "";
* });
* *todoList.addEventListener*("click", (e) => {
* // *check if user clicked on done button*
* if (*e.target.classList.contains*("remove")) {
* const *targetedLi* = *e.target.parentNode.parentNode*;
* *targetedLi.remove*();
* }
* if (*e.target.classList.contains*("done")) {
* const *liSpan* = *e.target.parentNode.previousElementSibling*;
* *liSpan.style.*textDecoration = "line-through";
* }
* });

# Asynchronous JavaScript

* Is Javascript a synchronous or asynchronous programming language ?
* // *synchronous programming vs asynchronous programming*
* // *synchronous programming*
* // *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**
* // *synchronous programming single threaded*
* //*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/*
* // *console.log("script start");*
* // *for (let i = 1; i < 10000; i++) {*
* // *console.log("inside for loop");*
* // *}*
* // *console.log("script end");*
* // *setTimeout return an id*
* *console.log*("script start");
* //*this function call after 1sec=1000*
* const *id* = *setTimeout*(()=>{
* *console.log*("inside setTimeout");
* },1000);
* *for* (leti=1; i < 100; i++) {
* *console.log*("....");
* }
* *console.log*("settimeout id is ", id);
* *console.log*("clearing time out");
* *clearTimeout*(id);// *to delete an id of setTimeout*
* *console.log*("Script end");
* SetTimeout()
* SetTimeout() with 0 millisecond
* Callback Queue
* SetInterval and create little project with setInterval
* // *setInterval - repeat fn after every interval*
* // *console.log("script start");*
* // *// setInterval(() => {*
* // *//   console.log(total);*
* // *//   console.log(Math.random());*
* // *// }, 500);*
* // *console.log("script end");*
* const *body* = *document.body*;
* const *button* = *document.querySelector*("button");
* const *intervalId* = *setInterval*(()=>{
* const *red* = *Math.floor*(*Math.random*()\*256);
* const *green* = *Math.floor*(*Math.random*()\*256);
* const *blue* = *Math.floor*(*Math.random*()\*256);
* const *rgb* =`*rgb(*${red}*,*${green}*,* ${blue}*)*`;
* *body.style.background* =rgb;
* },1000);
* *button.addEventListener*("click", () => {
* *clearInterval*(intervalId);
* *button.*textContent = *body.style.*background;
* });
* *console.log*(intervalId);
* Understand callbacks in general
* // *understand callback*
* // *function myFunc(callback) {*
* // *console.log("Function is doing task 1 ");*
* // *callback();*
* // *}*
* // *myFunc(() => {*
* // *console.log("function is doing task 2");*
* // *});*
* function *getTwoNumbersAndAdd*(number1, number2, onSuccess, onFailure) {
* if (typeof number1 === "number" && typeof number2 === "number") {
* *onSuccess*(number1, number2);
* } else {
* *onFailure*();
* }
* }
* function *addTwoNumbers*(num1, num2) {
* *console.log*(num1 + num2);
* }
* function *onFail*() {
* *console.log*("Wrong data type");
* *console.log*("please pass numbers only")
* }
* *getTwoNumbersAndAdd*(4, 4, addTwoNumbers, onFail);
* Callbacks in asynchronous programming
* Callback Hell and Pyramid of doom
* // *callbacks , callback hell, pyramid of doom*
* // *asynchronous programming*
* const *heading1* = *document.querySelector*(".heading1");
* const *heading2* = *document.querySelector*(".heading2");
* const *heading3* = *document.querySelector*(".heading3");
* const *heading4* = *document.querySelector*(".heading4");
* const *heading5* = *document.querySelector*(".heading5");
* const *heading6* = *document.querySelector*(".heading6");
* const *heading7* = *document.querySelector*(".heading7");
* const *heading8* = *document.querySelector*(".heading8");
* const *heading9* = *document.querySelector*(".heading9");
* const *heading10* = *document.querySelector*(".heading10");
* // *Text       Delay   Color*
* // *one        1s      Violet*
* // *two        2s      purple*
* // *three      2s      red*
* // *four       1s      Pink*
* // *five       2s      green*
* // *six        3s      blue*
* // *seven      1s      brown*
* // *callback hell*
* // *setTimeout(()=>{*
* // *heading1.textContent = "one";*
* // *heading1.style.color = "violet";*
* // *setTimeout(()=>{*
* // *heading2.textContent = "two";*
* // *heading2.style.color = "purple";*
* // *setTimeout(()=>{*
* // *heading3.textContent = "three";*
* // *heading3.style.color = "red";*
* // *setTimeout(()=>{*
* // *heading4.textContent = "four";*
* // *heading4.style.color = "pink";*
* // *setTimeout(()=>{*
* // *heading5.textContent = "five";*
* // *heading5.style.color = "green";*
* // *},2000)*
* // *},1000)*
* // *},2000)*
* // *},2000)*
* // *},1000)*
* function *changeText*(element, text, color, time, onSuccessCallback, onFailureCallback) {
* *setTimeout*(()=>{
* if(element){
* *element.*textContent = text;
* *element.style.*color = color;
* if(onSuccessCallback){
* *onSuccessCallback*();
* }
* }else{
* if(onFailureCallback){
* *onFailureCallback*();
* }
* }
* },time)
* }
* // *pyramid of doom*
* *changeText*(heading1, "one","violet",1000,()=>{
* *changeText*(heading2, "two","purple",2000,()=>{
* *changeText*(heading3, "three","red",1000,()=>{
* *changeText*(heading4, "four","pink",1000,()=>{
* *changeText*(heading5, "five","green",2000,()=>{
* *changeText*(heading6, "six","blue",1000,()=>{
* *changeText*(heading7, "seven","brown",1000,()=>{
* *changeText*(heading8, "eight","cyan",1000,()=>{
* *changeText*(heading9, "nine","#cda562",1000,()=>{
* *changeText*(heading10, "ten","dca652",1000,()=>{
* },()=>{*console.log*("Heading10 does not exist")})
* },()=>{*console.log*("Heading9 does not exist")})
* },()=>{*console.log*("Heading8 does not exist")})
* },()=>{*console.log*("Heading7 does not exist")})
* },()=>{*console.log*("Heading6 does not exist")})
* },()=>{*console.log*("Heading5 does not exist")})
* },()=>{*console.log*("Heading4 does not exist")})
* },()=>{*console.log*("Heading3 does not exist")})
* },()=>{*console.log*("Heading2 does not exist")})
* },()=>{*console.log*("Heading1 does not exist")})
* Intro to promises
* // *Promise*
* *console.log*("script start");
* const *bucket* =['coffee','chips','vegetables','salt','rice'];
* const *friedRicePromise* =new *Promise*((*resolve*, *reject*)=>{
* if(*bucket.includes*("vegetables")&& *bucket.includes*("salt")&& *bucket.includes*("rice")){
* *resolve*({ *value*:"friedrice"});
* }else{
* *reject*("could not do it");
* }
* })
* // *produce*
* // *consume*
* // *how to consume*
* *friedRicePromise.then*(
* // *jab promise resolve hoga*
* (myfriedRice) => {
* *console.log*("lets eat ", myfriedRice);
* }
* )*.catch*(
* (error) => {
* *console.log*(error)
* })
* *setTimeout*(() => {
* *console.log*("hello from settimeout")
* }, 0)
* *for* (leti=0; i <= 100; i++) {
* *console.log*(*Math.random*(), i);
* }
* *console.log*("script end!!!!")
* Microtask Queue
* Function that returns promise
* // *function returning promise*
* function *ricePromise*(){
* const *bucket* =['coffee','chips','vegetables','salts','rice'];
* *return* new Promise((resolve,reject)=>{
* if(*bucket.includes*("vegetables")&& *bucket.includes*("salt") && *bucket.includes*("rice")){
* *resolve*({value:"friedrice"});
* }else{
* *reject*("could not do it");
* }
* })
* }
* *ricePromise*()*.then*(
* // *jab promise resolve hoga*
* (myfriedRice)=>{
* *console.log*("lets eat ", myfriedRice);
* }
* )*.catch*(
* (error)=>{
* *console.log*(error)
* })
* Promise and settimeout
* // *promise && setTimeout*
* // *I want to resolve / reject promise after 2 seconds*
* function *myPromise*(){
* *return* new Promise((resolve, reject)=>{
* const *value* = *false*;
* *setTimeout*(()=>{
* if(value){
* *resolve*();
* }else{
* *reject*();
* }
* },2000)
* })
* }
* *myPromise*()
* *.then*(()=>{*console.log*("resolved")})
* *.catch*(()=>{*console.log*("rejected")})
* Promise.resolve and more about then method
* // *Promise.resolve*
* // *Promise chaining*
* // *const myPromise = Promise.resolve(5);*
* // *Promise.resolve(5).then(value=>{*
* // *console.log(value);*
* // *})*
* // *then()*
* // *then method hamesha promise return karta hai*
* function *myPromise*(){
* *return* new Promise((resolve, reject)=>{
* *resolve*("foo");
* })
* }
* *myPromise*()
* *.then*((value)=>{
* *console.log*(value);
* value += "bar";
* *return* value
* })
* *.then*((value) =>{
* *console.log*(value);
* value += "baaz";
* *return* value;
* })
* *.then*(value=>{
* *console.log*(value);
* })
* Convert nested Callbacks to ﬂat code using promises
* const *heading1* = *document.querySelector*(".heading1");
* const *heading2* = *document.querySelector*(".heading");
* const *heading3* = *document.querySelector*(".heading3");
* const *heading4* = *document.querySelector*(".heading4");
* const *heading5* = *document.querySelector*(".heading5");
* const *heading6* = *document.querySelector*(".heading6");
* const *heading7* = *document.querySelector*(".heading7");
* const *heading8* = *document.querySelector*(".heading8");
* const *heading9* = *document.querySelector*(".heading9");
* const *heading10* = *document.querySelector*(".heading10");
* function *changeText*(element, text, color, time) {
* *return* new Promise((resolve, reject) => {
* *setTimeout*(()=>{
* if(element){
* *element.*textContent = text;
* *element.style.*color = color;
* *resolve*();
* }else{
* *reject*("element not found");
* }
* },time)
* })
* }
* *changeText*(heading1, "one", "red", 1000)
* *.then*(()=>*changeText*(heading2, "two", "purple", 1000))
* *.then*(()=>*changeText*(heading3, "three", "green", 1000))
* *.then*(()=>*changeText*(heading4, "four", "orange", 1000))
* *.then*(()=>*changeText*(heading5, "four", "orange", 1000))
* *.then*(()=>*changeText*(heading6, "four", "orange", 1000))
* *.then*(()=>*changeText*(heading7, "four", "orange", 1000))
* *.then*(()=>*changeText*(heading8, "four", "orange", 1000))
* *.then*(()=>*changeText*(heading9, "four", "orange", 1000))
* *.then*(()=>*changeText*(heading10, "four", "orange", 1000))
* *.catch*((error)=>{
* *alert*(error);
* })
* Intro to Ajax, HTTP Request
* // *BASIC THEORY*
* // *AJAX : asynchronous javascript and XML*
* // *HTTP request*
* // *is a set of "web development techniques"*
* // *using many web technologies on the "client-side "*
* // *to create asynchronous web applications.*
* // *With Ajax, web applications can send and retrieve*
* // *data from a server asynchronously (in the background)*
* // *without interfering with the display and*
* // *behaviour of the existing page*
* // *We don't use data in XML format anymore.*
* // *we use JSON now.*
* // *we have 3 most common ways to create and send request to server*
* // *1.) xmlHTTPRequest (old way of doing)*
* // *2.) fetch API (new way of doing)*
* // *3.) axios (this is third party library)*
* XHR requests
* const *URL* ="https://jsonplaceholder.typicode.com/posts";
* const *xhr* =new *XMLHttpRequest*();
* // *console.log(xhr);*
* // *step1*
* // *console.log(xhr.readyState);*
* *xhr.open*("GET",URL);
* // *console.log(xhr.readyState);*
* // *xhr.onreadystatechange = function(){*
* // *// console.log(xhr.readyState);*
* // *if(xhr.readyState === 4){*
* // *console.log(xhr)*
* // *const response = xhr.response;*
* // *const data = JSON.parse(response);*
* // *console.log(typeof data);*
* // *}*
* // *}*
* *xhr.onload* = function(){
* const *response* = *xhr.response*;
* const *data* = *JSON.parse*(response);
* *console.log*(data);
* }
* *xhr.send*();
* Error handling in XHR requests
* const *URL* ="https://jsonplaceholder.typicode.com/posts";
* const *xhr* =new *XMLHttpRequest*();
* *xhr.open*("GET", URL);
* *xhr.onload* = () => {
* if(*xhr.*status >= 200 && *xhr.*status < 300) {
* const *data* = *JSON.parse*(*xhr.response*);
* *console.log*(data);
* const *id* =data[3]*.id*;
* const *xhr2* =new *XMLHttpRequest*();
* const *URL2* =`${*URL*}*/*${id}`
* *console.log*(URL2);
* *xhr2.open*("GET", URL2);
* *xhr2.onload* = () => {
* const *data2* = *JSON.parse*(*xhr2.response*);
* *console.log*(data2);
* }
* *xhr2.send*();
* }
* else{
* *console.log*("something went wrong");
* }
* }
* *xhr.onerror* = () => {
* *console.log*("network error");
* }
* *xhr.send*();
* XHR request Chaining
* Promisifying XHR requests and chaining using then method
* const *URL* ="https://jsonplaceholder.typicode.com/posts";
* function *sendRequest*(method, url) {
* *return* new Promise(function(resolve, reject) {
* const *xhr*=new *XMLHttpRequest*();
* *xhr.open*(method, url);
* *xhr.onload* = function() {
* if(*xhr.*status >= 200 && *xhr.*status < 300){
* *resolve*(*xhr.*response);
* }
* else{
* *reject*(new *Error*("Something Went wrong"));
* }
* }
* *xhr.onerror* = function() {
* *reject*(new *Error*("Something went wrong"));
* }
* *xhr.send*();
* })
* }
* *sendRequest*("GET", URL)
* *.then*(response => {
* const *data* = *JSON.parse*(response);
* // *console.log(data)*
* *return* data;
* })
* *.then*(data=>{
* const *id* =data[3]*.id*;
* *return* id;
* })
* *.then*(id=>{
* const *url* =`${*URL*}*/*${id}*ssss*`;
* *return* *sendRequest*("GET", url);
* })
* *.then*(newResponse => {
* const *newData* = *JSON.parse*(newResponse);
* *console.log*(newData);
* })
* *.catch*(error =>{
* *console.log*(error);
* })
* Fetch API
* // *fetch*
* const *URL* ="https://jsonplaceholder.typicode.com/postssss";
* *fetch*(URL,{
* method: 'POST',
* body: JSON*.stringify*({
* title: 'foo',
* body: 'bar',
* userId: 1,
* }),
* headers: {
* 'Content-type': 'application/json; charset=UTF-8',
* },
* })
* *.then*(response =>{
* if(*response.*ok){
* *return* *response.json*()
* }else{
* *throw* new *Error*("Something went wrong!!!")
* }
* })
* *.then*(data =>{
* *console.log*(data);
* })
* *.catch*(error =>{
* *console.log*("inside catch");
* *console.log*(error);
* })
* Error Handling in Fetch API
* Consume Promises with async and Await
* // *async await*
* // *fetch(URL)*
* // *.then(response => {*
* // *return response.json()*
* // *})*
* // *.then(data => {*
* // *console.log(data);*
* // *})*
* *console.log*("script start");
* const *URL* ="https://jsonplaceholder.typicode.com/posts";
* // *async function getPosts(){*
* // *const response = await fetch(URL);*
* // *if(!response.ok){*
* // *throw new Error("Something went wrong")*
* // *}*
* // *const data = await response.json();*
* // *return data;*
* // *}*
* const *getPosts* = *async*()=>{
* const *response* = *await fetch*(*URL*);
* if(!*response.ok*){
* *throw* new *Error*("Something went wrong")
* }
* const *data* = *await response.json*();
* *return* data;
* }
* // *const myData = getPosts();*
* // *console.log(myData);*
* *getPosts*()
* *.then*((myData) => {
* *console.log*(myData);
* })
* *.catch*(error =>{
* *console.log*("inside catch")
* *console.log*(error);
* })
* *console.log*("script end ");
* Split code into multiple ﬁles using ES6 modules.
* Congratulations
* Now you know javascript in Great Details

Thapa Technical 16hr javascript Source code Notes

console.log('console added')

*/\*\*\*\*  Section 1👉 we need to do it in console \*\*\*\*/*

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

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

 const chalk = require("chalk");

*/\*\*\*\* Section 2👉 Code Editor for writing JS \*\*\*\*/*

 var myName = 'vinod bahadur thapa';

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

 var myAge = 26;

 console.log(myAge);

 var \_myName = "vinod";

 Naming Practice

 var 1myName = "thapa"; not valid

 var \_1my\_\_Name = "bahadur";

 var $myName = "thapa technical";

 var myNaem% = "thapa technical";

 console.log(myNaem%);

  👉  🤩  https:www.youtube.com/channel/UCwfaAHy4zQUb2APNOGXUCCA

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  👉  🤩 SUBSCRIBE TO THAPA TECHNICAL YOUTUBE CHANNEL 🤩

 var myName = "vinod thapa";

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

 console.log(myName);

 var myAge = 26;

 console.log(myAge);

 var iAmThapas = false;

 console.log(iAmThapas);

  typeof operator

 console.log(typeof(iAmThapas));

 9 - "5"

 console.log( 9 - "5"); bug  4

 DataTypes Practice

 "Java" + "Script"

 console.log( 10 + "20"); 1020

 console.log( "Java "+ "Script");

 " " + " "

 console.log( " " + 0);

 " " + 0

 "vinod" - "thapa"

 true + true 1+1=1

 true + false 1+0=1

 false + true 0+1=1

 false - true 0-1=-1 true

 console.log("vinod" - "thapa");

 Difference between null vs undefined?

 🙋‍👨‍🏫 Interview Question 1 🙋‍👨‍🏫

 console.log(iAmUseless);

 console.log(typeof(iAmUseless));

 2nd javascript bug

 var iAmUseless = null;

 var iAmStandBy;

 console.log(iAmStandBy);

 console.log(typeof(iAmStandBy));

 What is NaN?

 NaN is a property of the global object.

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

 🙋‍👨‍🏫 Interview Question 2 🙋‍👨‍🏫

 The initial value of NaN is Not-A-Number

 var myName = "thapa technical";

 console.log(isNaN(myPhoneNumber));

 console.log(isNaN(myName));

 var myPhoneNumber = 9876543210;

 if(isNaN(myName)){

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

 }

 NaN Practice 🤯

 Number.NaN === NaN; false

 isNaN(NaN);         true

 isNaN(Number.NaN);  true

 NaN === NaN;        false

 Number.isNaN(NaN);  treu

 🙋‍👨‍🏫 Interview Question 1 🙋‍👨‍🏫

 console.log(NaN === NaN); true

 var vs let vs const

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

 1️⃣Assignment operators

 console.log(5+20);

 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 );

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

 I will tell you when we will see es6

 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);

 first using the original value of the variable and then the

 variable is incremented(increased).

 Postfix increment operator means the expression is evaluated

 the increment operator increments and returns the value after incrementing.

 var num = 15;

 var newNum = --num + 5;

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

 console.log(num);

 console.log(newNum);

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

 Prefix increment operator means the variable is incremented first and then

 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.

 Logical OR (||)

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

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

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

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

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

 takes truth to falsity and vice versa.

 console.log(!((a>0) || (b<0)));

 Logical NOT (!)

 console.log(!true);

  👉  🤩  https:www.youtube.com/channel/UCwfaAHy4zQUb2APNOGXUCCA

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  👉  🤩 SUBSCRIBE TO THAPA TECHNICAL YOUTUBE CHANNEL 🤩

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

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

 5️⃣ String Concatenation(operators)

 console.log("Hello World");

 console.log(myName + " bahadur");

 console.log("hello " +  "world");

 console.log(myName + " bahadur Thapa");

 var myName = "vinod";

 console.log(myName + " thapa");

 😳 4 Challenge Time

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

 console.log(5 + "thapa");

 sol 3: ✔

 sol 2:  ✔

 var b = 10;

 output b=5; a=10

 var a = 5;

 b = a;  b = 5;

 a = c;

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

 var c = b; c = 10

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

 var b = 10;

  output b=5; a=10

 sol 4: ✔

 var a = 5;

 b = a - b;  b = 5;

 a = a - b;  a = 10;

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

 a = a + b;  a = 15

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

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

 🙋‍👨‍🏫 Interview Question 4 🙋‍👨‍🏫

 var num1 = 5;

 var num2 = '5';

 console.log(typeof(num1));

 sol

 console.log(typeof(num2));

 console.log(num1 == num2 );

 var num2 = '5';

 console.log(typeof(num1));

 console.log(typeof(num2));

 var num1 = 5;

 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.

 else no raincoat

 if raining = raincoat

 var tomr = 'sunny';

 if(tomr == 'rain'){   console.log('take a raincoat'); }else{   console.log('No need to take a raincoat');

 }

 🤩Challenge Time

 write a program that works out whether if a given year is a leap year or not?

 A normal year has 365 days, leap years have 366, with an extra day in February.

 var year = 2020;

 debugger;

 if(year % 4 === 0){

   if(year % 100 === 0){

     if(year % 400 === 0){

       console.log("The year " + year + " is a leap year");

     }else{

       console.log("The year " + year + " is not a leap year");

     }

   }else{

     console.log("The year " + year + " is a leap year");

   }

 }else{

   console.log("The year " + year + " is not a leap year");

 }

 What is truthy and falsy values in Javascript?

 we have total 5 falsy values in javascript

 👉 0,"",undefined,null,NaN,false\*\* is false anyway

 if(score = 5){

   console.log("OMG, we loss the game 😭");

 }else{

   console.log("Yay, We won the game 😀");

 }

 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)

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  👉  🤩 SUBSCRIBE TO THAPA TECHNICAL YOUTUBE CHANNEL 🤩

  👉  🤩  https:www.youtube.com/channel/UCwfaAHy4zQUb2APNOGXUCCA

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 🙋‍👨‍🏫 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 );

 6️⃣ Anonymous Function

 A function expression is similar to and has the same syntax

 as a function declaration One can define "named"

 function expressions (where the name of the expression might

 be used in the call stack for example)

 or "anonymous" function expressions.

 var funExp = function(a,b){

   return total = a+b;

 }

 var sum = funExp(15,15);

 var sum1 = funExp(20,15);

 console.log(sum > sum1 );

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 👻 Now It's Time for Modern JavaScript 😍😍

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

 1️⃣ LET VS CONST  vs  VAR

 var myName = "thapa technical";

 console.log(myName);

 myName = "vinod thapa";

 console.log(myName);

 let myName = "thapa technical";

 console.log(myName);

 myName = "vinod thapa";

 console.log(myName);

 const myName = "thapa technical";

 console.log(myName);

 myName = "vinod thapa";

 console.log(myName);

 function biodata() {

   const myFirstName = "Vinod";

   console.log(myFirstName);

   if(true){

     const myLastName = "thapa";

   }

    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 = ['vinod', 'thapa', 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 : 'vinod',

     myLname : 'thapa',

     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 = "vinod";

       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 = "vinod thapa";

   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 vinod";

 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 = ['vinod','ramesh','arjun','vishal'];

 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 = ['vinod','ramesh','arjun','vishal'];

 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 = ['vinod','ramesh','arjun','vishal'];

 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 = ['vinod','ramesh','arjun','vishal'];

 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 = ["vinod","bahadur","thapa","thapatechnical","thapa"];

 console.log(myFriendNames.indexOf("Thapa", 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 = ["vinod","bahadur","thapa","thapatechnical","thapa"];

 console.log(myFriendNames.lastIndexOf("Thapa",3));

 Array.prototype.includes() 🙋‍♂️

 Determines whether the array contains a value,

 returning true or false as appropriate.

 var myFriendNames = ["vinod","bahadur","thapa","thapatechnical"];

 console.log(myFriendNames.includes("thapa"));

 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 )  );

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  👉  🤩 SUBSCRIBE TO THAPA TECHNICAL YOUTUBE CHANNEL 🤩

  👉  🤩  https:www.youtube.com/channel/UCwfaAHy4zQUb2APNOGXUCCA

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 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.

 😀7: challenge Time  🏁

 1: How to Sort the numbers in the array in ascending (up) and descending (down) order?

 compareFunction  Optional.

 A function that defines an alternative sort order. The function should return a negative, zero, or positive value, depending on the arguments, like:

 function(a, b){return a-b}

 for asecnding order

 array1.sort(function(a,b){

     console.log(a,b);

     if(a>b){

         return 1;

          b comes first and then a

     }

     if(a<b){

          a comes first and then b

         return -1;

     }

     if(a==b){

          No changes

         return 0;

     }

 });

 for desecnding order

 array1.sort(function(a,b){

     console.log(a,b);

     if(a>b){

         return -1;

          b comes first and then a

     }

     if(a<b){

          a comes first and then b

         return 1;

     }

     if(a==b){

          No changes

         return 0;

     }

 });

 console.log(array1);

 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

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  👉  🤩 SUBSCRIBE TO THAPA TECHNICAL YOUTUBE CHANNEL 🤩

  👉  🤩  https:www.youtube.com/channel/UCwfaAHy4zQUb2APNOGXUCCA

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 😀9: challenge Time 🏁

  1: Find the square root of each element in an array?

  2: Multiply each element by 2 and return only those

     elements which are greater than 10?

 sol1:

 let arr = [25, 36, 49, 64, 81];

 let arrSqr = arr.map((curElem) =>  Math.sqrt(curElem) )

 console.log(arrSqr);

 sol 2:

  let arr = [2, 3, 4, 6, 8];

  let arr2 = arr.map((curElm) => curElm \* 2).filter((curElem) => curElem > 10 ).reduce((accumulator, curElem) => {

       return accumulator += curElem;

     });

  console.log(arr2);

 we can use the chaining too

 👉 Reduce Method

 flatten an array means to convert the 3d or 2d array into a

 single dimensional array

 The reduce() method executes a reducer function (that you provide)

 on each element of the array, resulting in single output value.

 The reducer function takes four arguments:

 Accumulator

 Current Value

 Current Index

 Source Array

 4 subj = 1sub= 7

 3dubj = [5,6,2]

 let arr = [5,6,2];

 let sum = arr.reduce((accumulator, curElem) => {

         debugger;

       return accumulator += curElem;

 },7)

 console.log(sum);

 How to fatten an array

 converting 2d and 3d array into one dimensional array

 const arr = [

         ['zone\_1', 'zone\_2'],

         ['zone\_3', 'zone\_4'],

         ['zone\_5', 'zone\_6'],

         ['zone\_7', ['zone\_7', ['zone\_7', 'zone\_8']]]

     ];

  let flatArr = arr.reduce((accum, currVal)  => {

            return accum.concat(currVal);

  })

 console.log(arr.flat(Infinity));

 console.log(flatArr);

 const arr = [ ['zone\_1', 'zone\_2'], ['zone\_3', ['zone\_1', 'zone\_2', ['zone\_1', 'zone\_2']]] ];

 console.log(arr.flat(3));

 console.log(arr);

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

 A JavaScript string is zero or more characters written inside quotes.

 JavaScript strings are used for storing and manipulating text.

 You can use single or double quotes

 Strings can be created as primitives,

 from string literals, or as objects, using the String() constructor

 let myName = "vinod thapa";

 let myChannelName = 'vinod thapa';

  let ytName = new String("Thapa Technical");

 let ytName = 'thapa technical';

 console.log(myName);

 console.log((ytName));

 👉 How to find the length of a string

 String.prototype.length 🙋‍♂️

 Reflects the length of the string.

 let myName = "vinod thapa";

 console.log(myName.length);

 👉 Escape Character

 let anySentence = "We are the so-called \"Vikings\" from the north.";

 console.log(anySentence);

  if you dont want to mess, simply use the alternate quotes

 let anySentence =  " We are the so-called 'Vikings' from the north. ";

 console.log(anySentence);

 👉 Finding a String in a String

 String.prototype.indexOf(searchValue [, fromIndex]) 🙋‍♂️

 The indexOf() method returns the index of (the position of) the first

  occurrence of a specified text in a string

 const myBioData =  'I am the thapa Technical';

 console.log(myBioData.indexOf("t", 6));

  JavaScript counts positions from zero.

  0 is the first position in a string, 1 is the second, 2 is the third ...

  String.prototype.lastIndexOf(searchValue [, fromIndex]) 🙋‍♂️

  Returns the index within the calling String object of the

  last occurrence of searchValue, or -1 if not found.

 const myBioData =  'I am the thapa Technical';

 console.log(myBioData.lastIndexOf("t", 6));

 👉 Searching for a String in a String

 String.prototype.search(regexp) 🙋‍♂️

 The search() method searches a string for a specified

 value and returns the position of the match

 const myBioData =  'I am the thapa Technical';

 let sData = myBioData.search("technical");

 console.log(sData);

 The search() method cannot take a second start position argument.

 👉 Extracting String Parts

 There are 3 methods for extracting a part of a string:

 slice(start, end)

 substring(start, end)

 substr(start, length)

 The slice() Method 🙋‍♂️

 slice() extracts a part of a string and returns the extracted part

 in a new string.

 The method takes 2 parameters: the start position,

 and the end position (end not included).

 var str = "Apple, Bananaa, Kiwi, mango";

  let res = str.slice(0,4);

 let res = str.slice(7);

 console.log(res);

 The slice() method selects the elements starting at the

 given start argument, and ends at, but does not include,

 the given end argument.

 Note: The original array will not be changed.

 Remember: JavaScript counts positions from zero. First position is 0.

 😀11: challenge Time 🏁

 Display only 280 characters of a string like the

 one used in Twitter?

 let myTweets  = "Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Why do we use it? ";

 let myActualTweet = myTweets.slice(0,280);

 console.log(myActualTweet);

 console.log(myActualTweet.length);

 The substring() Method 🙋‍♂️

 substring() is similar to slice().

 The difference is that substring() cannot accept

 negative indexes.

 var str = "Apple, Bananaa, Kiwi";

 let res = str.substring(8,-2);

 console.log(res);

  If we give negative value then the characters are

  counted from the 0th pos

 The substr() Method 🙋‍♂️

 substr() is similar to slice().

 The difference is that the second parameter specifies the

 length of the extracted part.

 var str = "Apple, Bananaa, Kiwi";

  let res = str.substr(7,-2);

 let res = str.substr(-4);

 console.log(res);

 👉 Replacing String Content()

 String.prototype.replace(searchFor, replaceWith) 🙋‍♂️

 The replace() method replaces a specified value

 with another value in a string.

 let myBioData = `I am vinod bahadur thapa vinod`;

 let repalceData = myBioData.replace('Vinod','VINOD');

 console.log(repalceData);

 console.log(myBioData);

 Points to remember

 1: The replace() method does not change the string

 it is called on.  It returns a new string.

 2: By default, the replace() method replaces only

 the first match

 3:By default, the replace() method is case sensitive.

 Writing VINOD (with upper-case) will not work

👉 Extracting String Characters

 There are 3 methods for extracting string characters:

 charAt(position)

 charCodeAt(position)

 Property access [ ]

 The charAt() Method 🙋‍♂️

 The charAt() method returns the character at a

 specified index (position) in a string

 let str = "HELLO WORLD";

 console.log(str.charAt(9));

 The charCodeAt() Method 🙋‍♂️

 The charCodeAt() method returns the unicode of the

 character at a specified index in a string:

 The method returns a UTF-16 code

  (an integer between 0 and 65535).

 The Unicode Standard provides a unique number for every

 character, no matter the platform, device, application,

 or language. UTF-8 is a popular Unicode encoding which

  has 88-bit code units.

 var str = "HELLO WORLD";

 console.log( str.charCodeAt(0) );

 😀12: challenge Time 🏁

 Return the Unicode of the last character in a string

 let str = "HELLO WORLD";

 let lastChar = str.length - 1;

 console.log(str.charCodeAt(lastChar));

 Property Access

 ECMAScript 5 (2009) allows property access [ ] on strings

 var str = "HELLO WORLD";

 console.log(str[1]);

👉  Other useful methods

 let myName = "vinod tHapa";

 console.log(myName.toUpperCase());

 console.log(myName.toLowerCase());

 The concat() Method 🙋‍♂️

 concat() joins two or more strings

 let fName = "vinod"

 let lName = "thapa"

 console.log(fName + lName );

 console.log(`${fName} ${lName}`);

 console.log(fName.concat(lName));

 console.log(fName.concat(" " ,lName));

 String.trim() 🙋‍♂️

 The trim() method removes whitespace from both

 sides of a string

 var str = "              Hello         World!            ";

 console.log(str.trim());

 Converting a String to an Array

 A string can be converted to an array with the

 split() method

 var txt = "a, b,c d,e";    String

 console.log(txt.split(","));            Split on commas

 console.log( txt.split(" "));           Split on spaces

 console.log(txt.split("|"));          Split on pipe

*/\*\*\*\* Section 8👉 Date and Time in JavaScript \*\*\*\*/*

 JavaScript Date objects represent a single moment in time in a

 platform-independent format. Date objects contain a Number

 that represents milliseconds since 1 January 1970 UTC.

 👉 Creating Date Objects

 There are 4 ways to create a new date object:

 new Date()

 new Date(year, month, day, hours, minutes, seconds, milliseconds)

  it takes 7 *arguments*

 new Date(milliseconds)

  we cannot avoid month section

 new Date(date string)

 new Date() 🙋‍♂️

 Date objects are created with the new Date() constructor.

 let currDate = new Date();

 console.log(currDate);

 console.log(new Date());

 console.log(new Date().toLocaleString());  9/11/2019, 1:25:01 PM

 console.log(new Date().toString());  Wed Sep 11 2019 13:25:01 GMT+0700 (GMT+07:00)

 Date.now() 🙋‍♂️

 Returns the numeric value corresponding to the current time—the number

 of milliseconds elapsed since January 1, 1970 00:00:00 UTC

 console.log(Date.now());

 new Date(year, month, ...) 🙋‍♂️

 7 numbers specify year, month, day, hour, minute, second,

 and millisecond (in that order)

 Note: JavaScript counts months from 0 to 11.

 January is 0. December is 11.

 var d = new Date(2021,0);

 console.log(d.toLocaleString());

 new Date(dateString) 🙋‍♂️

 new Date(dateString) creates a new date object from a date string

 var d = new Date("October 13, 2021 11:13:00");

 console.log(d.toLocaleString());

 new Date(milliseconds) 🙋‍♂️

 new Date(milliseconds) creates a new date object as zero time plus milliseconds:

 var d = new Date(0);

 var d = new Date(1609574531435);

 var d = new Date(86400000\*2);

 console.log(d.toLocaleString());

👉 Dates Method

 const curDate = new Date();

  how to get the indivisual date

 console.log(curDate.toLocaleString());

 console.log(curDate.getFullYear());

 console.log(curDate.getMonth());  0-11 jan to dec

 console.log(curDate.getDate());

 console.log(curDate.getDay());

  how to set the indivisual date

 console.log(curDate.setFullYear(2022));

  The setFullYear() method can optionally set month and day

 console.log(curDate.setFullYear(2022, 10, 5));

 let setmonth = curDate.setMonth(10);  0-11 jan to dec

 console.log(setmonth);

 console.log(curDate.setDate(5));

 console.log(curDate.toLocaleString());

👉 Time Methods

 const curTime = new Date();

 how to get the indivisual Time

 console.log(curTime.getTime());

   The getTime() method returns the number of milliseconds

  since January 1, 1970

 console.log(curTime.getHours());

   The getHours() method returns the hours of a date as a

  number (0-23)

 console.log(curTime.getMinutes());

 console.log(curTime.getSeconds());

 console.log(curTime.getMilliseconds());

  how to set the indivisual Time

 let curTime = new Date();

  console.log(curTime.setTime());

 console.log(curTime.setHours(5));

 console.log(curTime.setMinutes(5));

 console.log(curTime.setSeconds(5));

 console.log(curTime.setMilliseconds(5));

  Practice Time

 new Date().toLocaleTimeString();  11:18:48 AM

 ---

 new Date().toLocaleDateString();  11/16/2015

 ---

 new Date().toLocaleString();  11/16/2015, 11:18:48 PM

 Challenge Time NOT yet decided

 (function(){

   setInterval(()=> {

     console.log(new Date().toLocaleTimeString());

   }, 1000)

 })();

*/\*\*\*\* Section 9👉 Math Object in JavaScript \*\*\*\*/*

 The JavaScript Math object allows you to perform mathematical tasks on numbers.

 console.log(Math.PI); 🙋‍♂️

 console.log(Math.PI);

 Math.round() 🙋‍♂️

 returns the value of x rounded to its nearest integer

 let num = 10.501;

 console.log(Math.round(num));

 Math.pow() 🙋‍♂️

 Math.pow(x, y) returns the value of x to the power of y

 console.log(Math.pow(2,3));

 console.log(2\*\*3);

 Math.sqrt() 🙋‍♂️

 Math.sqrt(x) returns the square root of x

 console.log(Math.sqrt(25));

 console.log(Math.sqrt(81));

 console.log(Math.sqrt(66));

 Math.abs() 🙋‍♂️

 Math.abs(x) returns the absolute (positive) value of x

 console.log(Math.abs(-55));

 console.log(Math.abs(-55.5));

 console.log(Math.abs(-955));

 console.log(Math.abs(4-6));

 Math.ceil() 🙋‍♂️

 Math.ceil(x) returns the value of x rounded up to its nearest integer

 console.log(Math.ceil(4.51));

 console.log(Math.round(4.51));

 console.log(Math.ceil(99.01));

 console.log(Math.round(99.1));

 Math.floor()  🙋‍♂️

 Math.floor(x) returns the value of x rounded down to its nearest integer

 console.log(Math.floor(4.7));

 console.log(Math.floor(99.1));

 Math.min() 🙋‍♂️

 Math.min() can be used to find the lowest value in a list of *arguments*

 console.log(Math.min(0, 150, 30, 20, -8, -200));

 Math.max() 🙋‍♂️

 Math.max() can be used to find the highest value in a list of *arguments*

 console.log(Math.max(0, 150, 30, 20, -8, -200));

 Math.random() 🙋‍♂️

 Math.random() returns a random number between 0 (inclusive), and 1 (exclusive)

 console.log(Math.floor(Math.random()\*10));

 console.log(Math.floor(Math.random()\*10));  0 to 9

 Math.round() 🙋‍♂️

 The Math.round() function returns the value of a number

 rounded to the nearest integer.

 console.log(Math.round(4.6));

 console.log(Math.round(99.1));

 Math.trunc()  🙋‍♂️

 The trunc() method returns the integer part of a number

 console.log(Math.trunc(4.6));

 console.log(Math.trunc(-99.1));

 Practice Time

 if the argument is a positive number, Math.trunc() is equivalent to

 Math.floor(),

 otherwise Math.trunc() is equivalent to Math.ceil().

 Section 10👉 Document Object model in JavaScript

 1️⃣ Window is the main container or we can say the

 global Object and any operations related to entire

 browser window can be a part of window object.

 For ex 👉 the history or to find the url etc.

 1️⃣ whereas the DOM is the child of Window Object

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  👉  🤩 SUBSCRIBE TO THAPA TECHNICAL YOUTUBE CHANNEL 🤩

  👉  🤩  https:www.youtube.com/channel/UCwfaAHy4zQUb2APNOGXUCCA

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 2️⃣ All the members like objects, methods or properties.

 If they are the part of window object then we do not refer

 the window object. Since window is the global object

 so you do not have to write down window.

 - it will be figured out by the runtime.

 For example

 👉 window.screen or just screen is a small information

 object about physical screen dimensions.

 👉 window.location giving the current URL

 👉 window.document or just document is the main object

 of the potentially visible (or better yet: rendered)

 document object model/DOM.

 2️⃣ Where in the DOM we need to refer the document,

 if we want to use the document object, methods or properties

 For example

 👉 document.getElementById()

 3️⃣ Window has methods, properties and object.

 ex setTimeout() or setInterval() are the methods

 where as Document is the object of the Window and

 It also has a screen object with properties

 describing the physical display.

 Now, I know you have a doubt like we have seen the methods

 and object of the global object that is window. But What about

 the properties of the Window Object 🤔

 so example of window object properties are

 innerHeight,

 innerWidth and there are many more

 let's see some practical in DOM HTML file

 \*\*\*\*\*\*\*\*\*\*\*\*\*\* DOM vs BOM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 👉 The DOM is the Document Object Model, which deals with the document,

 the HTML elements themselves, e.g. document and all traversal you

 would do in it, events, etc.

 For Ex: 👨‍🏫

 change the background color to red

 document.body.style.background = "red";

 👉 The BOM is the Browser Object Model, which deals with browser components

 aside from the document, like history, location, navigator and screen

 (as well as some others that vary by browser). OR

 In simple meaning all the Window operations which comes under BOM are performed

 usign BOM

 Let's see more practical on History object

 Functions alert/confirm/prompt are also a part of BOM:

 they are directly not related to the document,

 but represent pure browser methods of communicating with the user.

 alert(location.href);  shows current URL

 if (confirm("Want to Visit ThapaTechnical?")) {

   location.href = "https:www.youtube.com/thapatechnical";  redirect the browser to another URL

 }

 Section 3️⃣: Navigate through the DOM

 1: document.documentElement

       returns the Element that is the root element of the document.

 2: document.head

 3: document.body

 4: document.body.childNodes (include tab,enter and whiteSpace)

   list of the direct children only

 5: document.children (without text nodes, only regular Elements)

 6: document.childNodes.length

 👉 Practice Time

 How to check whether an element has child nodes or not?

 we will use hasChildNodes()

 👉 Practice Time

 How to find the child in DOM tree

 firstChild vs firstElementChild

 lastChild vs lastElementChild

 const data = document.body.firstElementChild;

 undefined

 data

 data.firstElementChild

 data.firstElementChild.firstElementChild

 data.firstElementChild.firstElementChild.style.color = "red"

 vs

 document.querySelector(".child-two").style.color = "yellow";

 👉 How to find the Parent Nodes

 document.body.parentNode

 document.body.parentElement

 👉 How to find or access the siblings

 document.body.nextSibling

 document.body.nextElementSibling

 document.body.previousSibling

 document.body.previousElementSibling

SECTION 4️⃣:  How to search the Elements and the References

 We will see the new file

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*/\*\*\*\* Section 11👉 EVENTS in JavaScript \*\*\*\*/*

 HTML events are "things" that happen to HTML elements.

 When JavaScript is used in HTML pages, JavaScript can "react" on these events.

 👨‍🏫 HTML Events

 An HTML event can be something the browser does, or something a user does.

 Here are some examples of HTML events:

 An HTML web page has finished loading

 An HTML input field was changed

 An HTML button was clicked

 Often, when events happen, you may want to do something.

 JavaScript lets you execute code when events are detected.

 HTML allows event handler attributes, with JavaScript code,

 to be added to HTML elements.

 section 1️⃣ 4 ways of writing Events in JavaScript

 1: using inline events alert();

 2: By Calling a funcion (We already seen and most common way of writing)

 3: using Inline events (HTML onclick="" property and element.onclick)

 4: using Event Listeners (addEventListener and IE's attachEvent)

 check the Events HTML File

 section 2️⃣: What is Event Object?

 Event object is the parent object of the event object.

 for Example

 MouseEvent, focusEvent, KeyboardEvent etc

 section ️3️⃣ MouseEvent in JavaScript

 The MouseEvent Object

 Events that occur when the mouse interacts with the HTML

 document belongs to the MouseEvent Object.

 section ️4️⃣ KeyboardEvent  in JavaScript

 Events that occur when user presses a key on the keyboard,

 belongs to the KeyboardEvent Object.

 https:www.w3schools.com/jsref/obj\_keyboardevent.asp

 Section 5️⃣ InputEvents in JavaScript

 The onchange event occurs when the value of an element has been changed.

 For radiobuttons and checkboxes, the onchange event occurs when the

 checked state has been changed.

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 👉  JavaScript Timing Events

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 The window object allows execution of code at specified time intervals.

 These time intervals are called timing events.

 The two key methods to use with JavaScript are:

 setTimeout(function, milliseconds)

 Executes a function, after waiting a specified number of milliseconds.

 setInterval(function, milliseconds)

 Same as setTimeout(), but repeats the execution of the function continuously.

 1️⃣ setTimeout()

 2️⃣clearTimeout()

 3️⃣ setInterval()

 4️⃣ clearInterval()

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 👉 object oriented Javascript

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 1️⃣ What is Object Literal?

 Object literal is simply a key:value pair data structure.

 Storing variables and functions together in one container,

 we can refer this as an Objects.

 object = school bag

 How to create an Object?

 1st way

   let bioData = {

       myName : "thapatechnical",

       myAge : 26,

       getData : function(){

         console.log(`My name is ${bioData.myName} and my age is ${bioData.myAge}`);

       }

   }

   bioData.getData();

 2nd way no need to write functions as well after es6

 let bioData = {

   myName : "thapatechnical",

   myAge : 26,

   getData (){

     console.log(`My name is ${bioData.myName} and my age is ${bioData.myAge}`);

   }

 }

 bioData.getData();

 👉 What if we want object as a value inside an Object

 let bioData = {

   myName : {

     realName : "vinod",

     channelName : "thapa technical"

   },

   myAge : 26,

   getData (){

     console.log(`My name is ${bioData.myName} and my age is ${bioData.myAge}`);

   }

 }

 console.log(bioData.myName.channelName );

 2️⃣ What is *this* Object?

 The definition  of "this" object is that it contain the current context.

 The *this* object can have different values depending on where it is placed.

  For Example 1

 console.log(*this*.alert('Awesome'));

  it refers to the current context and that is window global object

  ex 2

 function myName() {

     console.log(*this*);

 }

 myName();

  ex 3

 var myNames = 'vinod';

 function myName() {

     console.log(*this*.myNames);

 }

 myName();

  ex 4

 const obj = {

     myAge : 26,

     myName() {

       console.log(*this*.myAge);

     }

 }

 obj.myName();

  ex 5

*this* object will not work with arrow function bcz arrow function is bound to class.

 const obj = {

     myAge : 26,

     myName : () => {

       console.log(*this*);

     }

 }

 obj.myName();

  ex 6

 let bioData = {

     myName : {

         realName : "vinod thapa",

         channelName : 'thapa technical'

     },

      things to remember is that the myName is the key and the object is act like a value

     myAge : 26,

     getData (){

       console.log(`My name is ${*this*.myName.channelName} and my age is ${*this*.myAge} `);

     }

   }

   bioData.getData();

  call method is used to call the method of another object

  or with call(), an object can use a method belonging to another object

  But as per other it is simply the way to use the this keyword or another object

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  👉 How JavaScript Works? Advanced and Asynchronous JavaScript

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  Advanced JavaScript Section

  1️⃣: Event Propagation (Event Bubbling and Event Capturing)

  check html file

  2️⃣: Higher Order Function

  function which takes another function as an arguments is called HOF

  wo function jo dusre function ko as an argument accept krta hai use HOF

  3️⃣: Callback Function

  function which get passed as an argument to another function is called CBF

  A callback function is a function that is passed as an argument to

  another function, to be “called back” at a later time.

  Jis bhi function ko hum kisi or function ke under as an arguments passed

  krte hai then usko hum CallBack fun bolte hai

   we need to create a calculator

 const add = (a,b) => {

     return a+b;

 }

  console.log(add(5,2));

 const subs = (a,b) => {

     return Math.abs(a-b);

 }

 const mult = (a,b) => {

     return a\*b;

 }

 const calculator = (num1,num2, operator) => {

   return operator(num1,num2);

 }

 calculator(5,2,subs)

 console.log(calculator(5,2,subs));

   I have to do the hardcoded for each operation which is bad

   we will use the callback and the HOF to make it simple to use

   Now instead of calling each function indivisually we can call it

   by simply using one function that is calculator

 console.log(calculator(5,6,add));

 console.log(calculator(5,6,subs));

 console.log(calculator(5,6,mult));

  In the above example, calculator is the higher-order function,

  which accepts three arguments, the third one being the callback.

  Here the calculator is called the Higher Order Function because it takes

  another function as an argument

  and add, sub and mult are called the callback function bcz they are passed

  as an argument to another fucntion

  InterView Question

  Difference Between Higher Order Function and Callback Function ?

  🏁🏁Asynchronous JavaScript

  6️⃣:   Synchronous JavaScript Prog

 1work = 10min

 2work = 5s

 const fun2 = () => {

   console.log(`Function 2️⃣ is called`);

 }

 const fun1 = () => {

   console.log(`Function 1 is called`);

   fun2();

   console.log(`Function 1 is called Again ✌`);

 }

 fun1();

 Asynchronous JavaScript Prog

 const fun2 = () => {

   setTimeout(()=> {

       console.log(`Function 2️⃣  is called`);

   }, 2000);

 }

 const fun1 = () => {

   console.log(`Function 1 is called`);

   fun2();

   console.log(`Function 1 is called Again ✌`);

 }

 fun1();

 👻 What is Event Loop in JavaScript?

  ppt explain

  5️⃣ Hoisting in JavaScript

  we have a creation phase and execution phase.

  Hoisting in Javascript is a mechanism where variables and functions

  declarations are moved to the top of their scope before the code execute.

 For Example 👇

 console.log(myName);

 let myName;

 myName = "thapa";

  How it will be in output during creation phase

 1: var myName = undefined;

 2: console.log(myName);

 3: myName = "thapa";

 😲 In ES2015 (a.k.a. ES6), hoisting is avoided by using the let keyword

  instead of var. (The other difference is that variables declared

  with let are local to the surrounding block, not the entire function.)

  6️⃣ What is Scope Chain and Lexical Scoping in JavaScript?

  The scope chain is used to resolve the value of variable names

  in JS.

  scope chain in js is lexically defined, which means that we can

  see what the scope chain will be by looking at the code.

  At the top, we have the Global Scope, which is the window Object

  in the browser.

  Lexical Scoping means Now, the inner function can get access to

  their parent functions variables But the vice-versa is not true.

  For Example 👇

 let a = "Hello guys. ";  global scope

 const first= () => {

   let b = " How are you?"

     const second = () => {

       let c = " Hii, I am fine thank you🙏";

       console.log(a+b+c);

     }

     second();

     console.log(a+b+c); I can't use C

 }

 first();

  7️⃣ What is Closures in JavaScript 🤔

  A closure is the combination of a function bundled together (enclosed) with references

  to its surrounding state (the lexical environment).

  In other words, a closure gives you

  access to an outer function’s scope from an inner function.

  In JavaScript, closures are created every time a function is created, at function creation time.

  For Example 👇

 const outerFun = (a) => {

     let b = 10;

     const innerFun = () => {

       let sum = a+b;

       console.log(`the sum of the two no is ${sum}`);

     }

     innerFun();

 }

 outerFun(5);

  it same like lexical scoping

  One more Example 👇

 const outerFun = (a) => {

   let b = 10;

   const innerFun = () => {

     let sum = a+b;

     console.log(`the sum of the two no is ${sum}`);

   }

   return innerFun;

 }

 let checkClousure = outerFun(5);

 console.dir(checkClousure);

 "use strict"

 let x = "vinod";

 console.log(x);

 🏁🏁🏁 Back To Advanced JavaScript

 Currying

 const sum = (num1) => (num2) => (num3) => console.log(num1+num2+num3);

 sum(5)(3)(8);

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  👉  🤩 SUBSCRIBE TO THAPA TECHNICAL YOUTUBE CHANNEL 🤩

  👉  🤩  https:www.youtube.com/channel/UCwfaAHy4zQUb2APNOGXUCCA

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  8️⃣: CallBack Hell

 setTimeout(()=>{

     console.log(`1️⃣ works is done`);

     setTimeout(()=>{

         console.log(`2️⃣ works is done`);

         setTimeout(()=>{

             console.log(`3️⃣ works is done`);

             setTimeout(()=>{

                 console.log(`4️⃣ works is done`);

                 setTimeout(()=>{

                     console.log(`5️⃣ works is done`);

                     setTimeout(()=>{

                         console.log(`6️⃣ works is done`);

                     }, 1000)

                 }, 1000)

             }, 1000)

         }, 1000)

     }, 1000)

 }, 1000)

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  👉  🤩 Bonus JSON 🤩

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  👉 JSON.stringify turns a JavaScript object into JSON text and

 stores that JSON text in a string, eg:

 var my\_object = { key\_1: "some text", key\_2: true, key\_3: 5 };

 var object\_as\_string = JSON.stringify(my\_object);

  "{"key\_1":"some text","key\_2":true,"key\_3":5}"

 console.log(object\_as\_string);

 typeof(object\_as\_string);

 "string"

  👉 JSON.parse turns a string of JSON text into a JavaScript object, eg:

 var object\_as\_string\_as\_object = JSON.parse(object\_as\_string);

  {key\_1: "some text", key\_2: true, key\_3: 5}

 typeof(object\_as\_string\_as\_object);

  "object"

  7️⃣ AJAX Call using XMLHttprequest

  how to handled with the events and callback

  XMLHttpRequest (XHR) objects are used to interact with servers.

  You can retrieve data from a URL without having to do a full

  page refresh. This enables a Web page to update just part

  of a page without disrupting what the user is doing.

  XMLHttpRequest is used heavily in AJAX programming.

 const request = new XMLHttpRequest();

  we need to call the api or request the api using GET method ki, me jo

  url pass kar kr rha hu uska data chaiye

 request.open('GET', "https:covid-api.mmediagroup.fr/v1");

 request.send();  we need to send the request and its async so we need to

  add the event to load the data adn get it

  to get the response

 request.addEventListener("load", () => {

     console.log(*this*.responseText);

 });