**Basic Of Programming in JavaScript**

1- Hello world program

// console.log can print something on console

console.log("hello World");

2- Declare variable using var

"use strict";

// => Restrict to do Mistake

// intro to variables

// variables can store some information

// we can use that information later

// we can change that information later

// declare a variable

var firstName = "Devesh";

// use a variable

console.log(firstName);

// change value

firstName = "Kumar";

console.log(firstName);

// Case Sensitive

3- More about variable

// rules for naming variables

// you cannot start with number

// example :-

// 1value (invalid)

// value1 (valid)

var value1 = 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

// first name (invalid)

// var first\_name = "harshit"; // snake case writing

// var firstName = "harshit"; // camel case writing

// convention

// start with small letter and use camelCase

4- 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)

5- Const

// declare constants

const pi = 3.14;

pi=3.15; // we cant change the value of const variable;

console.log(pi);

6- String Indexing

// String indexing

let firstName = "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

7- Useful string methods

// trim()

// toUpperCase()

// toLowerCase()

// slice()

let firstName = "harshit";

// String is immutable.

// console.log(firstName.length);=> use to find the length of string

// firstName = firstName.trim(); // "harshit" =>trim all the spaces

// console.log(firstName)

// console.log(firstName.length);

// firstName = firstName.toUpperCase();=> Convert all character into uppercase

// firstName = firstName.toLowerCase();=> Convert all character into lower case

// console.log(firstName);

// start index

// end index

let newString = firstName.slice(1); // hars

newString=firstName.slice(2,5);

console.log(newString);

8- Template Strings

// string concatenation

let string1 = "17";

let string2 = "10";

let newString = +string1 + +string2;

console.log(typeof newString);

// template string

let age = 22;

let firstName = "harshit"

// "my name is harshit and my age is 22 "

// let aboutMe = "my name is " + firstName + " and my age is " + age;

let aboutMe = `my name is ${firstName} and my age is ${age}`

console.log(aboutMe);

9- Null, undefined, BigInt, typeof

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

// convert number to string. 22 -> "22"

age = age + "";

console.log(typeof(age)); "22"

// convert string to number.

let myStr = +"34";

console.log(typeof myStr);

// let age = "18";

// age = Number(age);

// console.log(typeof age);

// undefined

// null

// BigInt

let firstName;

console.log(typeof firstName); // firstname is undefined

firstName = "Harshit";

console.log(typeof firstName, firstName);

let myVariable = null;

console.log(myVariable);

// myVariable = "Devesh";

console.log(myVariable, typeof myVariable); // Devesh,"string".

console.log(typeof null); // bug , error

// BigInt

let myNumber = BigInt(12);

let sameMyNumber = 123n;

// console.log(myNumber);

// console.log(Number.MAX\_SAFE\_INTEGER);

console.log(myNumber+ sameMyNumber);

10- Booleans and Comparison Operator

// booleans & comparison operator

// booleans -> true, false

let num1 = 7;

let num2 = "7";

//Comparison Operator

console.log(num1<num2);

// == vs ===

console.log(num1 === num2);  // this will check value and datatype

// != vs !==

console.log(num1 !== num2);

11- Truthy and Falsy Values

// false

// ""

// null

// undefined

// 0

// truthy

// "abc"

// 1, -1

12- If else statement

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");

}

//-------------------------------------//

let firstName= 0;

if(firstName){

    console.log(firstName);

}else{

    console.log("firstName is kinda empty");

}

13- 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);

14-&& || operator

// and  or operator

let firstName = "arshit";

let age = 16;

if(firstName[0] === "H"){

    console.log("your name starts with H")

}

if(age > 18){

    console.log("you are above 18");

}

// AND Operator

if(firstName[0] === "H" && age>18){

    console.log("Name starts with H and above 18");

}else{

    console.log("inside else");

}

// OR Operator

if(firstName[0] === "H" || age>18){

    console.log("inside if");

}else{

    console.log("inside else");

}

15- Nested if else

// nested if else

// winning number 19

// 19 your guess is right

// 17 too low

// 20 too high

let winningNumber = 19;

let userGuess = +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 !!!");

    }

}

16- 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");

17- Switch statement

// switch statement

let day = 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");

}

18- While loop

// while loop

// 0 se 9

// dry don't repeat yourself

let i = 0; // 1 2 3 4

while(i<=9){

    console.log(i);

    i++;

}

console.log(`current value of i is ${i}`);

console.log("hello");

19- While loop examples

// while loop example

let num = 100;

let total = 0;

let i = 0;

while(i<=100){

    total = total + i;

    i++;

}

console.log(total);

let total = (num\*(num+1))/2;

console.log(total);

20- For Loop

// intro to for loop

// print 0 to 9

for(let i = 0;i<=9;i++){

    console.log(i);

}

console.log("value of i is ",i);

21-For loop Example

// for loop example

let total = 0;

let num = 100;

for(let i = 1; i<=num; i++){

    total = total + i;

}

console.log(total);

22- Break and continue keyword

// break keywork

for(let i = 1; i<=10; i++){

    if(i===4){

        break;

    }

    console.log(i);

}

// continue keyword

for(let i = 1; i<=10; i++){

    if(i===4){

        continue;

    }

    console.log(i);

}

console.log("hello there");

23- 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);

**Array in JavaScript**

24-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]);

let fruits = ["apple", "mango", "grapes"];

let obj = {}; // object literal

// console.log(fruits);

// fruits[1] = "banana";

// console.log(fruits);

console.log(typeof fruits);

console.log(typeof obj);

console.log(Array.isArray(fruits));

console.log(Array.isArray(obj));

// array indexing

25- Push pop shift unshift

// array push pop

// array shift unshift

let fruits = ["apple", "mango", "grapes"];

console.log(fruits);

push

fruits.push("banana");

console.log(fruits);

pop

let poppedFruit = fruits.pop();

console.log(fruits);

console.log("popped fruits is", poppedFruit);

unshift

fruits.unshift("banana");

fruits.unshift("myfruit");

console.log(fruits);

shift

let removedFruit = fruits.shift();

console.log(fruits);

console.log("removed fruits is ", removedFruit);

26- Primitive vs reference data types

// primitve vs reference data types

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

let array1 = ["item1", "item2"];

let array2 = 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);

27- Clone array & spread operator

// how to clone array

// how to concatenate two arrays

let array1 = ["item1", "item2"];

let array2 = ["item1", "item2"];

let array2 = array1.slice(0).concat(["item3", "item4"]);

let array2 = [].concat(array1,["item3", "item4"]);

// new way

// spread operator

let oneMoreArray = ["item3", "item4"]

let array2 = [...array1, ...oneMoreArray];

array1.push("item3");

console.log(array1===array2);

console.log(array1)

console.log(array2)

28- For loop

// for loop in array

let fruits = ["apple", "mango", "grapes", "banana"];

for(let i=0; i<=9;i++){

    console.log(i);

}

// console.log(fruits.length);

// console.log(fruits[fruits.length-2]);

let fruits2 = [];

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

    fruits2.push(fruits[i].toUpperCase());

}

console.log(fruits2);

29- 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);

30- While loop in array

// while loop in array

const fruits = ["apple", "mango", "grapes"];

const fruits2 = [];

let i = 0;

while(i<fruits.length){

    fruits2.push(fruits[i].toUpperCase());

    i++;

}

console.log(fruits2);

31- For of loop

// for of loop in array

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]);

}

32- For in loop

// for in loop in array

const fruits = ["apple", "mango", "grapes", "fruit4", "fruit5"];

const fruits2 = [];

for(let index in fruits){

    fruits2.push(fruits[index].toUpperCase());

}

console.log(fruits2);

33- 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);

Object In JavaScript

34- 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,

    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";

console.log(person);

35- 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);

36- Iterate objects

// how to iterate object

const person = {

    name: "harshit",

    age: 22,

    "person hobbies": ["guitar", "sleeping", "listening music"]

}

// for in loop

// Object.keys

for(let key in person){

    // console.log(`${key} : ${person[key]}`);

    console.log(key," : " ,person[key]);

}

console.log(typeof (Object.keys(person)));

const val = Array.isArray((Object.keys(person)));

console.log(val);

for(let key of Object.keys(person)){

    console.log(person[key]);

}

37- 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);

38- 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"];

// console.log(newArray);

// spread operator in objects

const obj1 = {

  key1: "value1",

  key2: "value2",

};

const obj2 = {

  key1: "valueUnique",

  key3: "value3",

  key4: "value4",

};

// const newObject = { ...obj2, ...obj1, key69: "value69" };

// const newObject = { ...["item1", "item2"] };

// const newObject = { ..."abcdefghijklmnopqrstuvwxyz" };

// console.log(newObject);

39- 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);

40- 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(let user of users){

    console.log(user.firstName);

}

41- 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 JacaScript

42- 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(let i = 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);

43- 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(let i = 0; i<array.length; i++){

        if(array[i]===target){

            return i;

        }

    }

    return -1;

}

44- 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(let i = 0; i<array.length; i++){

        if(array[i]===target){

            return i;

        }

    }

    return -1;

}

45- Function declarations are hoisted

// hoisting

// hello();

// function hello(){

//     console.log("hello world");

// }

// console.log(hello);

// const hello = "hello world";

// console.log(hello);

46- 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();

47-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();

48- 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){

        var firstName = "harshit";

        console.log(firstName);

    }

    if(true){

        console.log(firstName);

    }

    console.log(firstName);

}

myApp();

49- 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);

50- 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){

    let total = 0;

    for(let number of numbers){

        total = total + number;

    }

    return total;

}

const ans = addAll(4,5,4,2,10);

console.log(ans);

51- 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);

52- Very brief intro to callback functions

// 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);

53- 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

54- Foreach method

// important array methods

 const numbers = [4,2,5,8];

function myFunc(number, index){

    console.log(`index is ${index} number is ${number}`);

}

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);

}

55-Map Method

// map method

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

// const square = function(number){

//     return number\*number;

// }

// 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);

56-Filter

// filter method

const numbers = [1,3,2,6,4,8];

const evenNumbers = numbers.filter((number)=>{

    return number % 2 === 0;

});

console.log(evenNumbers);

57-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);

// 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

58-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;

// });

// numbers.sort((a,b)=>a-b);

// console.log(numbers);

// 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);

59-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);

60-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);

61-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);

62-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);

63-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

64-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]);

65-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);

66-Maps

// Maps

// 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));

// 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);

67- 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);

68- Optional chaining

// optional chaining

const user  = {

    firstName: "harshit",

    // address: {houseNumber: '1234'}

}

console.log(user?.firstName);

console.log(user?.address?.houseNumber);

Object Oriented JavaScript

69- 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();

70- This keyword, Window object

// console.log(window);

// "use strict";

// function myFunc(){

//     console.log(this);

// }

// myFunc();

71- 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,

}

// apply

// about.apply(user1, ["guitar", "bach"]);

// const func = about.bind(user2, "guitar", "bach");

// func();

72- 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.bind(user1);

myFunc();

73- This inside arrow functions

// arrow functions

const user1 = {

    firstName : "harshit",

    age: 8,

    about: () => {

        console.log(this.firstName, this.age);

    }

}

user1.about(user1);

74- 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();

75- 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);

76- 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;

}

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());

77- 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());

78- 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

obj2.key3 = "value3";

// obj2.key2 = "unique";

console.log(obj2);

console.log(obj2.\_\_proto\_\_);

79- What is prototype

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());

80- 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());

81-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());

82- 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(let key in user1){

    // console.log(key);

    if(user1.hasOwnProperty(key)){

        console.log(key);

    }

}

83- 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");

// }

84- 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));

85- 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());

86- 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());

87- 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());

88- 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);

89- 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 Work**

91-Global Execution Context & This and Window

// 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);

var firstName = "Harshit";

console.log(firstName);

92-Hoisting

// hoisting

console.log(this);

console.log(window);

console.log(myFunction);

console.log(fullName);

function myFunction(){

    console.log("this is my function");

}

var firstName = "Harshit";

var lastName = "Sharma"

var fullName = firstName + " " + lastName;

console.log(fullName);

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");

let firstName = "Harshit";

let lastName = "Vashistha";

const myFunction = function() {

    let var1 = "First Variable";

    let var2 = "second Variable";

    console.log(var1);

    console.log(var2);

}

93- Function execution context

// function execution context

let foo = "foo";

console.log(foo);

function getFullName(firstName, lastName){

    console.log(arguments);

    let myVar = "var inside func";

    console.log(myVar);

    const fullName = firstName + " " + lastName;

    return fullName;

}

const personName = getFullName("harshit", "sharma");

console.log(personName);

94-

// lexical environment, scope chain

const lastName = "Vashistha";

const printName = function(){

    const firstName = "harshit";

    function myFunction(){

        console.log(firstName);

        console.log(lastName);

    }

    myFunction()

}

printName();

95-

// 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();

96-

function hello(x){

    const a  = "varA";

    const b = "varB";

    return function(){

        console.log(a,b,x);

    }

}

const ans = hello("arg");

ans();

97-

// 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);

98-

function func(){

    let counter = 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

99- HTML and CSS Crash course

100- Async vs defer

101- Select elements using id

102- querySelector

103- textContent & innerText

104-Change the styles of elements using js

105- Get and set attributes

106- Select multiple elements and loop through them

107- innerHTML

108- Deeply understand dom tree, root node , element nodes, text nodes

109- classList

110-Add new elements to page

111- Create elements

112- Insert adjacent elements

113- Clone nodes

114- More methods to add elements on page

115- How to get the dimensions of the element

116- Intro to events

117- This keyword inside eventListener callback

118- Add events on multiple elements

119- Event object

120- How event listener works

121- Practice with events

122- Create demo project

123- More events

124- Event bubbling

125- Event Capturing

126- Event delegation

127- Create Project using event delegation