ES6 Assignments

1. **Constants:**Declare a constant & confirm its value cannot be changed.

const a=10;

a=20;

console.log(a);

//Gives error assignment to constant variable

1. **Scoping:**Declare a variable inside if condition & make sure that it is not accessible outside if condition.

if(true){

    let a=10;

}

console.log(a);

//error a is not defined

1. **Enhanced object properties:**Create an ‘Order’ object having data members ‘id’, ‘title’, ‘price’. Add the methods printOrder() &getPrice(). Now, copy the order object using Object.assign().

let user = {

    id : 1,

    title : "order",

    price : 100,

    printOrder(){

        return "id : " + this.id + " \ntitle: " + this.title + " \nprice " + this.price;

    }

    getPrice(){

        return "price: " + this.price;

    }

};

let obj = Object.assign(user);

console.log(obj);

1. **Arrow functions:**Take an array of strings & convert it into another array of object which has two properties {string, string\_length}. For example:

let names = [‘Tom’, ‘Ivan’, ‘Jerry’]

Output: [ {name: ’Tom’, length: 3}, {name: ’Ivan’, length: 4 }, {name: ’Jerry’, length: 5} ]

let names = [‘"Tom", "Ivan", "Jerry"];

let array=[]

for(let i of names){

     array.push({name:i;length:i.length;})

}

console.log(array)

1. **Extended parameter handling:**
   1. Write a add() with default values.

let add=(a,b)=>{

     return a+b;

}

console.log(add(5,15));

* 1. Write a function userFriends() that takes 2 arguments username & array of user friends. The function should print username & his list of friends. (Use rest parameters)

function userFriends(username,...userfriend){

     return username+" : "+userfriend;

}

console.log(userFriends("pk","sk","rk","ay","kj"));

* 1. Write a function printCapitalNames() that takes five names as argument & prints them in capital letters. Use spread operator in order to call printCapitalNames() function.

let names=["pk","sk","rk","ay","kj"]

function printCapitalNames(a,b,c,d,e){

     return a+" "+b+" "+c+" "+d+" "+e;

}

 console.log(printCapitalNames(...names));

1. **Template literals:**Draft a ticket to Sysnet that describes problem with your laptop. Use ‘template literals’ to add value of laptop model, your desk no, your name etc.

let laptop = {

    model: 2021,

    desk\_no: 1,

    name: "PK",

};

function getTicket() {

    console.log(`Mr ${laptop.name}, Your laptop model: ${laptop.model} is assigned to desk no ${laptop.desk\_no}. Description : Screen is Broken.`);

}

getTicket();

1. **De-structuring assignment:**
   1. Suppose there is a javascript array with 4 elements. Print the value of 3rd element using array matching.

let arr: number[] = [1, 2, 3, 4];

let [a, b, c, d] = arr;

console.log(c);

* 1. Create an organization object having attributes name, address. Write a program to retrieve pin code of an address using object deep matching.

let myOrg = {

   name: "org",

   address: {

     city: "org city",

     country: "India",

     pin: 415002,},};

let {

   address: { pin },

} = myOrg;

console.log(pin);

1. **Classes & Modules:**Write a class Account with attributes id, name, balance. Add two sub classes SavingAccount&CurrentAccount having specific attribute interest &cash\_credit respectively. Create multiple saving & current account objects. Write a functionality to find out total balance in the bank.

class Account {

  Id: string;

  Name: string;

  Balance: number;

  constructor(id: string, name: string, balance: number) {

    this.Id = id;

    this.Name = name;

    this.Balance = balance;

  }

  getBalance() {

    return this.Balance;

  }

}

let t=0;

class SavingsAccount extends Account {

  Interest: number;

  Cash\_credit: number;

  constructor(id: string, name: string, interest: number, cash\_credit: number) {

    super(id, name, cash\_credit);

    this.Interest = interest;

    this.Cash\_credit = cash\_credit;

    t += cash\_credit;

  }

}

class CurrentAccount extends Account {

  Interest: number;

  Cash\_credit: number;

  constructor(id: string, name: string, interest: number, cash\_credit: number) {

    super(id, name, cash\_credit);

    this.Interest = interest;

    this.Cash\_credit = cash\_credit;

    t += cash\_credit;

  }

}

let Savacc1 = new SavingsAccount("1", "SK", 10, 1000);

let Savacc2 = new SavingsAccount("2", "PK", 10, 2000);

let Curracc1 = new CurrentAccount("3", "RK", 10, 3000);

function getTotalBalance() {

  console.log(t);

}

getTotalBalance();