MODULE: 2

(Data Types and Objects)

1). Write the code, one line for each action:

Ans.

1. Create an empty object user.

const user = {};

This code declares a new constant called "user" and assigns an empty object to it using curly braces **{}**. You can now add properties and methods to this object as needed.

b) Add the property name with the value John.

user.name = "John";

This code uses dot notation to add a new property called "name" to the "user" object and sets its value to "John". Now the "user" object has a property called "name" with the value of "John".

c) Add the property surname with the value Smith.

user.surname = "Smith";

This code uses dot notation to add a new property called "surname" to the "user" object and sets its value to "Smith". Now the "user" object has two properties: "name" with the value of "John" and "surname" with the value of "Smith".

d) Change the value of the name to Pete.

user.name = "Pete";

This code uses dot notation to access the "name" property of the "user" object and sets its value to "Pete". Now the "user" object has two properties: "name" with the value of "Pete" and "surname" with the value of "Smith".

e) Remove the property name from the object.

delete user.name;

This code uses the **delete** keyword to remove the "name" property from the "user" object. Now the "user" object has only one property: "surname" with the value of "Smith".

2) Is array copied? let fruits = ["Apples", "Pear", "Orange"]; // push a new value into the "copy" let shoppingCart = fruits; shoppingCart.push("Banana"); // what's in fruits?

Ans.

alert(fruits.length);In the given code snippet, the variable **shoppingCart** is assigned a reference to the same array that **fruits** is pointing to. Therefore, when a new value "Banana" is pushed into **shoppingCart**, it will also be added to **fruits**.

   <script>

    let fruits = ["Apples", "Pear", "Orange"];

let shoppingCart = fruits;

shoppingCart.push("Banana");

console.log(fruits);

</script>

3) Map to names let john = { name: "John", age: 25 }; let pete = { name: "Pete", age: 30 }; let mary = { name: "Mary", age: 28 }; let users = [ john, pete, mary ]; let names = /\* ... your code \*/ alert( names ); // John, Pete, Mary

Ans.

   <script>

    let john = { name: "John", age: 25 };

let pete = { name: "Pete", age: 30 };

let mary = { name: "Mary", age: 28 };

let users = [john, pete, mary];

let names = users.map(function(user) {

  return user.name;

});

alert(names); // output: John, Pete, Mary

   </script>

This code declares three objects **john**, **pete**, and **mary** with a **name** and **age** property. It also declares an array called **users** and assigns the three objects to it.

The **map** method is called on the **users** array with a callback function that takes each object **user** as input and returns the value of its **name** property. The **map** method then creates a new array **names** with these returned values.

Finally, the **alert** method is used to display the **names** array, which contains the names "John", "Pete", and "Mary".

4) Sum the properties There is a salaries object with arbitrary number of salaries. Write the function sumSalaries(salaries) that returns the sum of all salaries using Object.values and the for..of loop.If salaries is empty, then the result must be 0. let salaries = {

"John": 100,

"Pete": 300,

"Mary": 250 }; alert( sumSalaries(salaries) ); // 650

Ans.

In this implementation, the **sumSalaries** function takes an object **salaries** as its parameter. It initializes a variable **sum** to **0**, then uses a **for...of** loop to iterate over the values in the **salaries** object (using the **Object.values** method), adding each salary to the **sum** variable. Finally, it returns the total sum.

The code then creates an object **salaries** with three key-value pairs, representing the salaries of three people. It passes this object as an argument to the **sumSalaries** function, and displays the result of the function (the sum of the salaries) using the **alert** function. If the **salaries** object were empty, the function would return **0**.

 <script>

function sumSalaries(salaries) {

  let sum = 0;

  for (let salary of Object.values(salaries)) {

    sum += salary;

  }

  return sum;

}

let salaries = {

  "John": 100,

  "Pete": 300,

  "Mary": 250

};

alert(sumSalaries(salaries)); // 650

    </script>

5). Destructuring assignment We have an object: Write the Destructuring assignment that reads: a) Name property into the variable name. b) Year’s property into the variable age. c) isAdmin property into the variable isAdmin (false, if no such property) d) let user = { name: "John", years: 30};

Ans.

<script>

     let user = {

        name: "John",

        years: 30

      };

      let { name, years: age, isAdmin = false } = user;

      console.log(name);     // "John"

      console.log(age);      // 30

      console.log(isAdmin);  // false

   </script>

6). Turn the object into JSON and back Turn the user into JSON and then read it back into another variable. user = { name: "John Smith", age: 35};

Ans.

<script>

        let user = {

  name: "John Smith",

  age: 35

};

// Convert the object to JSON format

let userJson = JSON.stringify(user);

// Parse the JSON string back into a new object

let newUser = JSON.parse(userJson);

console.log(user);      // { name: "John Smith", age: 35 }

console.log(userJson);  // {"name":"John Smith","age":35}

console.log(newUser);   // { name: "John Smith", age: 35 }

    </script>

**Thank you**