Advance JavaScript

MODULE: 2 (Data Types and Objects)

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

a) Create an empty object user.

Ans : let user={};

b) Add the property name with the value John

Ans : User.name=”Rahul”

c) Add the property surname with the value Smith.

Ans : user.surname = "Patel";

d) Change the value of the name to Pete.

Ans : user.name = "Bitu";

e) Remove the property name from the object.

Ans : delete user.name;

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? alert(fruits.length); // ?

Ans : The result is 4:

Input:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      let fruits = ["Apples", "Pear", "Orange"];

      let shoppingCart = fruits;

      shoppingCart.push("Banana");
      alert(fruits.length); // 4
    </script>
  </body>
</html>
```

Output:

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.
Input:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
</head>
<body>
   <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(item => item.name);

alert( names ); // John, Pete, Mary
   </script>
</body>
</html>
```

Output:

4. Map to objects let john = { name: "John", surname: "Smith", id: 1 }; let pete = { name: "Pete", surname: "Hunt", id: 2 }; let mary = { name: "Mary", surname: "Key", id: 3 }; let users = [john, pete, mary]; let usersMapped = /* ... your code ... */
Ans.
Input:
Example 1:

```
<!DOCTYPE html>
<html lang="en">
  <head>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      let john = { name: "John", surname: "Smith", id: 1 };
      let pete = { name: "Pete", surname: "Hunt", id: 2 };
      let mary = { name: "Mary", surname: "Key", id: 3 };

      let users = [john, pete, mary];

      let usersMapped = users.map((user) => ({
        fullName: `${user.name} ${user.surname}`,
        id: user.id,
      }));


// usersMapped = [
//   { fullName: "John Smith", id: 1 },
//   { fullName: "Pete Hunt", id: 2 },
//   { fullName: "Mary Key", id: 3 }
// ]

      alert(usersMapped[0].id); // 1
      alert(usersMapped[0].fullName); // John Smith
    </script>
  </body>
</html>
```

Output:

Example 2:
Input:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
```

```
    <script>
    //   let john = { name: "John", surname: "Smith", id: 1 };
    //   let pete = { name: "Pete", surname: "Hunt", id: 2 };
    //   let mary = { name: "Mary", surname: "Key", id: 3 };

    //   let users = [john, pete, mary];

    //   let usersMapped = users.map((user) => ({
    //     fullName: `${user.name} ${user.surname}`,
    //     id: user.id,
    //   }));


usersMapped = [
  { fullName: "John Smith", id: 1 },
  { fullName: "Pete Hunt", id: 2 },
  { fullName: "Mary Key", id: 3 }
]

      alert(usersMapped[0].id); // 1
      alert(usersMapped[0].fullName); // John Smith
    </script>
  </body>
</html>
```

Output:

5. 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.

Input:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
```

```
<body>
    <script>
        function sumSalaries(salaries) {

let sum = 0;
for (let salary of Object.values(salaries)) {
  sum += salary;
}

return sum; // 650
}

let salaries = {
"John": 100,
"Pete": 300,
"Mary": 250
};

alert( sumSalaries(salaries) ); // 650

    </script>
</body>
</html>
```

Output:

6 . 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.

Input:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
```

```
    let user = {
      name: "John",
      years: 30,
    };

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

    alert(name); // John
    alert(age); // 30
    alert(isAdmin); // false
  </script>
 </body>
</html>
```

Output:

7. 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 :

```
let user = {
name: "John Smith",
age: 35
};
let user2 = JSON.parse(JSON.stringify(user));
```