MODULE: 2 (Data Types and Objects)

• Write the code, one line for each action:

a) Create an empty object user.

b) Add the property name with the value John.

c) Add the property surname with the value Smith.

d) Change the value of the name to Pete.

e) Remove the property name from the object.

Ans

Let user = {};

User .name = “John”;

User. surname = “smith”;

User.name = “pete”;

Delete User.name;

● Is array copied?

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

// push a new value into the "copy" let shopping Cart = fruits; shopping Cart. push("Banana");

// what's in fruits? Alert (fruits .length); // ?

Ans

The result is 4:

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

let shopping Cart = fruits;

shopping Cart. push("Banana");

alert (fruits. length); // 4

That’s because arrays are objects. So both shopping Cart and fruits are the references to the same array.

● 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

```
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
```

● 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 ... */ /*
```

```
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
```

Ans

```
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
```

● 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( sum Salaries (salaries) ); // 650
```

Ans

```
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 (sum Salaries (salaries)); // 650
```

● 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) is Admin property into the variable is Admin (false, if no such property)

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

Ans

```
Let user = {
Name: “john”,
Years: 30
};
Let {name, years: age, is Admin = false} = user;
Alert(name); // John
Alert(age); //30
Alert (is Admin); //false
```

● 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
};
Lets user2 = JSON.parse(JSON.stringify(user));
```