**MODULE :-Advance javascript & Essentials**

**Q.1 Write a program to Show an alert**

ANS: - Alert(“Alert Message”)

**Q.2What will be the result for these expressions?**

**1. 5 > 4**

**2. "apple" > "pineapple"**

**3. "2" > "12"**

**4. undefined == null**

**5. undefined === null**

**6. null == "\n0\n" 7.**

**7. null === +"\n0\n"**

ANS. 1 5 > 4

Result: true

2. "apple" > "pineapple"

Result: false

3. "2" > "12"

Result: true

4. undefined == null

Result: true

5. undefined === null

Result: false

6. null == "\n0\n"

Result: false

7. null === +"\n0\n"

Result: false

**Q.3 Will alert be shown?**

**if ("0") { alert( 'Hello'); }**

ANS. Yes, the alert will be shown.

**Q.4What is the code below going to output?**

alert( null || 2 || undefined );

ANS. The code will output 2.

**Data type and object**

**Q.1 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 // a) Create an empty object user.

const user = {};

// b) Add the property name with the value John.

user.name = "John";

// c) Add the property surname with the value Smith.

user.surname = "Smith";

// d) Change the value of the name to Pete.

user.name = "Pete";

// e) Remove the property name from the object.

delete user.name;

**Q.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. Yes, the array is not copied in this case. The variable shoppingCart is referencing the same array object as the variable fruits, so any modifications made to shoppingCart will also affect the original fruits array

Code :-

console.log(fruits); // Output: ["Apples", "Pear", "Orange", "Banana"]

console.log(shoppingCart); // Output: ["Apples", "Pear", "Orange", "Banana"]

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

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

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

let users = [john, pete, mary];

// Use map() to extract names from each object in the users array

let names = users.map(user => user.name);

alert(names); // Output: "John, Pete, Mary"

**Q.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 ... \*/ /\* 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];**

**// Use map() to create a new array of objects with the fullName property**

**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); // Output: 1**

**alert(usersMapped[0].fullName); // Output: 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); // Output: 1

alert(usersMapped[0].fullName); // Output: John Smith

**Q.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. function sumSalaries(salaries) {

let sum = 0;

const salaryValues = Object.values(salaries);

for (let salary of salaryValues) {

sum += salary;

}

return sum;

}

let salaries = {

"John": 100,

"Pete": 300,

"Mary": 250

};

alert(sumSalaries(salaries)); // Output: 650

**Q.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. let user = { name: "John", years: 30 };

// a) Name property into the variable name.

let { name } = user;

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

// b) Year's property into the variable age.

let { years: age } = user;

console.log(age); // Output: 30

// c) isAdmin property into the variable isAdmin (false, if no such property)

let { isAdmin = false } = user;

console.log(isAdmin); // Output: false (since the isAdmin property does not exist in the user object)

**Q.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. let user = { name: "John Smith", age: 35 };

// Step 1: Turn the user object into JSON

let userJSON = JSON.stringify(user);

// userJSON will be a string containing the JSON representation of the user object:// '{"name":"John Smith","age":35}'

// Step 2: Read the JSON back into another variable as an object

let userCopy = JSON.parse(userJSON);

// userCopy is now a separate object with the same properties as the original user object:

// { name: "John Smith", age: 35 }

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

**New Request**

**Q.1 What is JSON**

Ans.JSON stands for JavaScript Object NotationJSON is a lightweight format for storing and transporting dataJSON is often used when data is sent from a server to a web pageJSON is "self-describing" and easy to understand

**Q.2 What is promises**

ANS. Promises are a feature in JavaScript used for handling asynchronous operations. Asynchronous operations are tasks that take some time to complete, such as fetching data from a server, reading a file, or making an API call. Instead of blocking the execution of code until the task is finished, promises allow you to define callbacks that will be executed once the asynchronous operation is completed or has encountered an error.

**Javascript Essentials**

**Q.1 What is JavaScript Output method?**

ANS. JavaScript can "display" data in different ways:

Writing into an HTML element, using innerHTML.

Writing into the HTML output using document.write().

Writing into an alert box, using window.alert().

Writing into the browser console, using console.log()

**Q.2 How to used JavaScript Output method?**

ANS. JavaScript can "display" data in different ways:

Alert (pop-up dialog):

Prompt (pop-up dialog for user input):

Confirm (pop-up dialog with OK and Cancel buttons):