**Advance JavaScript**

**MODULE: 1 (Introduction and Code Quality)**

Q1. Write a program to Show an alert

Ans. <script>

function showAlert()

{ alert("Hello, World!"); }

</script>

Q2. What 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. null === +"\n0\n"

Ans 1. True

2. False

3. True

4. True

5. False

6. False

7. False

Q3. Will alert be shown?

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

Ans. Yes alert function() will be shown.

Q4. What is the code below going to output? alert( null || 2 || undefined );

Ans. 2

Q5. The following function returns true if the parameter age is greater than 18. Otherwise it asks for a confirmation and returns its result:

function

checkAge(age)

{

if (age> 18) { return true; }

else { // ...return confirm (‘did parents allow you?');

}

}

Q6. Replace Function Expressions with arrow functions in the code below: Function

ask(question, yes, no)

{ if (confirm(question))yes();

else

no();

}

ask("Do you agree?", function()

{ alert("You agreed."); },

function() {

alert("You canceled the execution."); }

}

Ans. const ask = (question, yes, no) => {

if (confirm(question)) {

yes();

} else {

no();

}

};

ask(

"Do you agree?",

() => {

alert("You agreed.");

},

() => {

alert("You canceled the execution.");

}

);

**MODULE: 2 (Data Types and Objects)**

Q1. 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. const user = {};

user.name = "John";

user.surname = "Smith";

user.name = "Pete";

delete user.name;

Q2. 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. In the given code, the shoppingCart variable is not a copy of the fruits array, but it is referencing the same array object as fruits. This means that any changes made to shoppingCart will also affect the fruits array, because they both point to the same array object in memory.

Therefore, after the code runs, the fruits array will contain the additional value "Banana" that was pushed onto shoppingCart. The alert(fruits.length) statement will output 4, because there are now four elements in the fruits array: ["Apples", "Pear", "Orange", "Banana"].

Q3. 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(user => user.name);

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

Q4. 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

Q5. 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;

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

sum += salary;

}

return sum;

}

let salaries = {

"John": 100,

"Pete": 300,

"Mary": 250

};

alert(sumSalaries(salaries)); // 650

Q6. 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 };

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

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

console.log(age); // 30

console.log(isAdmin); // false

Q7. 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 userJson = JSON.stringify(user);

console.log(userJson);

let userObject = JSON.parse(userJson);

console.log(userObject);

**MODULE: 3 (Document, Event and Controls)**

Q1. Create a program to hide/show the password

Ans. <!DOCTYPE html>

<html>

<head>

<title>Show/Hide Password Example</title>

<script>

function togglePasswordVisibility() {

var passwordInput = document.getElementById("password");

var toggleCheckbox = document.getElementById("toggle");

if (toggleCheckbox.checked) {

passwordInput.type = "text";

} else {

passwordInput.type = "password";

}

}

</script>

</head>

<body>

<label>

Password:

<input id="password" type="password" />

</label>

<br />

<label>

Show Password:

<input id="toggle" type="checkbox" onclick="togglePasswordVisibility()" />

</label>

</body>

</html>

Q2. Create a program that will select all the classes and loop over and whenever i click the button the alert should show

Ans. <!DOCTYPE html>

<html>

<head>

<title>Click button to show alert</title>

</head>

<body>

<button id="my-button">Click me!</button>

<script>

const elements = document.querySelectorAll('.my-class');

elements.forEach(element => {

element.addEventListener('click', () => {

alert('You clicked an element with class "my-class"!');

});

});

const button = document.querySelector('#my-button');

button.addEventListener('click', () => {

alert('You clicked the button!');

});

</script>

</body>

</html>

Q3. Create a responsive header using proper JavaScript

Ans. <!DOCTYPE html>

<html>

<head>

<title>Responsive Header Example</title>

<style>

.header {

background-color: blue;

color: white;

padding: 10px;

}

.header.large {

background-color: red;

color: yellow;

padding: 20px;

}

</style>

</head>

<body>

<header class="header">Header</header>

<script>

window.matchMedia('(min-width: 768px)').addListener(handleScreenSizeChange);

function handleScreenSizeChange(e) {

if (e.matches) {

document.querySelector('.header').classList.add('large');

} else {

document.querySelector('.header').classList.remove('large');

}

}

</script>

</body>

</html>

Q4. Create a form and validate using JavaScript

Ans. <!DOCTYPE html>

<html>

<head>

<title>Form Validation</title>

</head>

<body>

<form id="myForm">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required>

<br>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<br>

<label for="password">Password:</label>

<input type="password" id="password" name="password" minlength="8" required>

<br>

<button type="submit">Submit</button>

</form>

<script>

const form = document.getElementById('myForm');

form.addEventListener('submit', (event) => {

event.preventDefault();

const name = document.getElementById('name').value;

const email = document.getElementById('email').value;

const password = document.getElementById('password').value;

if (name === '' || email === '' || password === '') {

alert('Please fill out all fields');

} else if (password.length < 8) {

alert('Password must be at least 8 characters long');

} else {

alert('Form submitted successfully!');

form.reset();

}

});

</script>

</body>

</html>

Q5. Create a modal box using css and Js with three buttons

Ans. <!DOCTYPE html>

<html>

<head>

<title>Modal Box Example</title>

<link rel="stylesheet" type="text/css" href="styles.css">

</head>

<body>

<h1>Modal Box Example</h1>

<button id="openModal">Open Modal</button>

<div id="modal" class="modal">

<div class="modal-content">

<h2>Modal Header</h2>

<p>Modal content goes here.</p>

<button id="closeModal">Close Modal</button>

</div>

</div>

<script src="script.js"></script>

</body>

</html>

**CSS:-**

.modal {

display: none;

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: rgba(0,0,0,0.5);

}

.modal-content {

background-color: white;

margin: 20% auto;

padding: 20px;

border: 1px solid black;

max-width: 500px;

text-align: center;

}

.modal-content button {

margin: 10px;

padding: 10px 20px;

border: none;

background-color: #333;

color: white;

font-size: 16px;

cursor: pointer;

}

.modal-content button:hover {

background-color: #555;

}

.modal-content h2 {

margin-top: 0;

}

button#openModal {

margin: 20px;

padding: 10px 20px;

border: none;

background-color: #333;

color: white;

font-size: 16px;

cursor: pointer;

}

button#openModal:hover {

background-color: #555;

}

**JS:-**

const openModalBtn = document.getElementById("openModal");

const closeModalBtn = document.getElementById("closeModal");

const modal = document.getElementById("modal");

openModalBtn.addEventListener("click", () => {

modal.style.display = "block";

});

closeModalBtn.addEventListener("click", () => {

modal.style.display = "none";

});

window.addEventListener("click", (event) => {

if (event.target === modal) {

modal.style.display = "none";

}

});

Q6. Use external js library to show slider

Ans. <!DOCTYPE html>

<html>

<head>

<head>

<link rel="stylesheet" type="text/css" href="path/to/slick.css"/>

<script src="path/to/jquery.min.js"></script>

<script src="path/to/slick.min.js"></script>

<script>

$(document).ready(function(){

$('.slider').slick({

autoplay: true,

autoplaySpeed: 3000,

dots: true,

arrows: false

});

});

</script>

</head>

</head>

<body>

<div class="slider">

<div>

<img src="path/to/image1.jpg">

</div>

<div>

<img src="path/to/image2.jpg">

</div>

<div>

<img src="path/to/image3.jpg">

</div>

</div>

</body>

Q7. Prevent the browser when i click the form submit button

Ans. <form id="myForm" action="submit-form.php" method="post">

<!-- form inputs and other elements here -->

<button type="submit" id="submitButton">Submit</button>

</form>

<script>

document.getElementById("myForm").addEventListener("submit", function(event) {

event.preventDefault();

});

</script>

**MODULE: 4 (New Request)**

Q1. What is JSON

Ans. JSON stands for JavaScript Object Notation. It is a lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate. JSON is often used to transmit data between a server and a web application, as an alternative to XML. JSON data is represented as a collection of key-value pairs, similar to a dictionary or object in JavaScript.

Q2. What is promises

Ans. Promises in JavaScript are a way of handling asynchronous operations. An asynchronous operation is one that doesn't complete immediately and may take some time to finish. In JavaScript, asynchronous operations are typically handled using callbacks. However, callbacks can lead to a situation called "callback hell", where nested callbacks make the code hard to read and maintain.

Promises provide a more elegant way to handle asynchronous operations. A promise represents a value that may not be available yet, but will be at some point in the future. It's a placeholder for a value that's being computed asynchronously.

Q3. Write a program of promises and handle that promises also

Ans. function fetchData() {

return new Promise(function(resolve, reject) {

// Perform an asynchronous operation, such as an API call or database query

// Here we'll simulate a delay using setTimeout and return some sample data

setTimeout(function() {

const data = { message: "Hello, World!" };

resolve(data); // Resolve the promise with the data

}, 2000); // Delay for 2 seconds

});

}

// Call the fetchData function and handle the promise

fetchData()

.then(function(data) {

// The promise was resolved successfully, handle the data

console.log(data.message); // Output: "Hello, World!"

})

.catch(function(error) {

// The promise was rejected with an error, handle the error

console.error("An error occurred: " + error);

})

.finally(function() {

// Perform cleanup or other tasks after the promise is settled (resolved or rejected)

console.log("Promise settled");

});

Q4. Use fetch method for calling an api https://fakestoreapi.com/products

Ans. fetch('https://fakestoreapi.com/products')

.then(response => response.json())

.then(data => {

console.log(data);

// Do something with the data returned from the API

})

.catch(error => {

console.error(error);

// Handle any errors that occur during the fetch operation

});

Q5. Display all the product from the api in your HTML page

Ans. <!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Product List</title>

</head>

<body>

<ul id="productList"></ul>

<script>

fetch('https://fakestoreapi.com/products')

.then(response => response.json())

.then(data => {

const productList = document.getElementById('productList');

data.forEach(product => {

const li = document.createElement('li');

li.textContent = `${product.title} - $${product.price}`;

productList.appendChild(li);

});

})

.catch(error => console.error(error));

</script>

</body>

</html>