**Q.1. Write a program to Show an alert**Ans. alert ('this is an alert');

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

Ans.

1. 5 > 4

Result: true

Explanation: 5 is greater than 4, so the expression is true.

2. "apple" > "pineapple"

Result: false

Explanation: In lexicographic (dictionary) order, "apple" comes before "pineapple," so the expression is false.

3. "2" > "12"

Result: true

Explanation: In lexicographic order, "2" comes after "12," so the expression is true.

4. undefined == null

Result: true

Explanation: In JavaScript, undefined and null are loosely equal (==) to each other, so the expression is true.

5. undefined === null

Result: false

Explanation: undefined and null are not strictly equal (===) to each other because they have different types, so the expression is false.

6. null == "\n0\n"

Result: false

Explanation: In loose equality (==), null is only equal to undefined, not to a string like "\n0\n," so the expression is false.

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

Result: false

Explanation: The strict equality (===) operator compares not only values but also types. null is of type "object," and +"\n0\n" converts the string to a number, which is 0. Since their types are different, the expression is false.

**Q.3. Will alert be shown? if ("0") {alert (‘Hello');}**

Yes, an alert will be shown.

In JavaScript, when you use a non-empty string (in this case, "0") in a conditional statement like an if statement, it is considered truthy. Therefore, the code inside the if block will be executed, and the alert('Hello') statement will display an alert with the message "Hello."

**Q.4. What is the code below going to output? alert (null || 2 || undefined);**

The code will provide output 2.

This is because the || (logical OR) operator in JavaScript returns the first truthy value it encounters from left to right. In this case, null is false, so it moves on to the next value, which is 2. 2 is a truthy value, so the expression short-circuits, and 2 is returned and displayed in the alert. undefined is not even evaluated in this case.

**Q.5. 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) {else { } } if (age> 18) {return true; } // ...return confirm (‘did parents allow you?');**

function checkAge(age) {

if (age > 18) {

return true;

} else {

const isAllowed = confirm ('Did your parents allow you?');

return isAllowed;

}

}

**Q.6.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."); } }**

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

if (confirm(question)) {

yes();

} else {

no();

}

};

ask(

"Do you agree?",

() => {

alert("You agreed.");

},

() => {

alert("You canceled the execution.");

}

);

**Data Types and Objects**

**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.**

a) const user = {};

b) user.name = "John";

c) user.surname = "Smith";

d) user.name = "Pete";

e) 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. In this code, when you do let shoppingCart = fruits;, you are not creating a copy of the fruits array. Instead, you are creating a new reference to the same array. Both fruits and shoppingCart will refer to the same array in memory. So, when you push "Banana" into shoppingCart, it will also affect the fruits array.

As a result, when you check the length of the fruits array using alert(fruits.length), it will be 4, as "Banana" has been added to the array.

**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];

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

alert(names); // "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];

let usersMapped = users.map(user => ({

fullName: `${user.name} ${user.surname}`,

id: user.id

}));

alert(usersMapped[0].id); // 1

alert(usersMapped[0].fullName); // "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 total = 0;

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

total += salary;

}

return total;

}

let salaries = {

"John": 100,

"Pete": 300,

"Mary": 250

};

alert(sumSalaries(salaries)); // 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;

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

let { years: age } = user;

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

let { isAdmin = false } = user;

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

console.log(age); // 30

console.log(isAdmin); // false

**Q.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 };

// Convert the user object to JSON

let userJSON = JSON.stringify(user);

// Parse the JSON string back into another variable

let newUser = JSON.parse(userJSON);

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

**Document, Event and Controls**

**Q.1. Create a program to hide/show the password?**

Ans. <!DOCTYPE html>

<html>

<head>

<title>Password Toggle Example</title>

</head>

<body>

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

<input type="password" id="password" placeholder="Enter your password">

<input type="checkbox" id="showPassword"> Show Password

<script>

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

const showPasswordCheckbox = document.getElementById("showPassword");

showPasswordCheckbox.addEventListener("change", function() {

if (showPasswordCheckbox.checked) {

passwordInput.type = "text"; // Show the password

} else {

passwordInput.type = "password"; // Hide the password

}

});

</script>

</body>

</html>

**Q.2. 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>Alert on Button Click</title>

</head>

<body>

<button id="showAlertButton">Show Alert</button>

<div class="alert-class">Element 1</div>

<div class="alert-class">Element 2</div>

<div class="alert-class">Element 3</div>

<script>

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

const alertElements = document.querySelectorAll(".alert-class");

showAlertButton.addEventListener("click", function() {

alertElements.forEach(function(element) {

alert("Alert from: " + element.textContent);

});

});

</script>

</body>

</html>

**Q.3. Create a responsive header using proper JavaScript?**

Ans. <!DOCTYPE html>

<html>

<head>

<title>Responsive Header Example</title>

<style>

/\* Add your CSS styles for the header and navigation here \*/

header {

background-color: #333;

color: #fff;

padding: 10px;

text-align: center;

}

nav {

display: none;

}

@media (min-width: 768px) {

nav {

display: block;

}

}

</style>

</head>

<body>

<header>

<h1>Responsive Header</h1>

<button id="toggleNavButton">Toggle Navigation</button>

</header>

<nav>

<ul>

<li><a href="#">Home</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Services</a></li>

<li><a href="#">Contact</a></li>

</ul>

</nav>

<script>

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

const nav = document.querySelector("nav");

toggleNavButton.addEventListener("click", function() {

if (nav.style.display === "none" || nav.style.display === "") {

nav.style.display = "block";

} else {

nav.style.display = "none";

}

});

</script>

</body>

</html>

**Q.4. Create a form and validate using JavaScript?**

Ans. <!DOCTYPE html>

<html>

<head>

<title>Form Validation Example</title>

<style>

.error {

color: red;

}

</style>

</head>

<body>

<h2>Form Validation Example</h2>

<form id="myForm" onsubmit="return validateForm()">

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

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

<span class="error" id="nameError"></span><br>

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

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

<span class="error" id="emailError"></span><br>

<input type="submit" value="Submit">

</form>

<script>

function validateForm() {

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

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

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

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

nameError.innerHTML = "";

emailError.innerHTML = "";

let valid = true;

if (name === "") {

nameError.innerHTML = "Name is required";

valid = false;

}

if (email === "") {

emailError.innerHTML = "Email is required";

valid = false;

} else if (!/^\S+@\S+\.\S+$/.test(email)) {

emailError.innerHTML = "Invalid email format";

valid = false;

}

return valid;

}

</script>

</body>

</html>

**Q.5. Create a modal box using css and Js with three buttons?**

Ans.

HTML:

<!DOCTYPE html>

<html>

<head>

<title>Modal Box Example</title>

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

</head>

<body>

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

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

<div class="modal-content">

<span class="close" id="closeModalBtn">&times;</span>

<h2>Modal Box</h2>

<p>This is a simple modal box with three buttons.</p>

<button id="button1">Button 1</button>

<button id="button2">Button 2</button>

<button id="button3">Button 3</button>

</div>

</div>

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

</body>

</html>

CSS:

/\* Modal Styles \*/

.modal {

display: none;

position: fixed;

z-index: 1;

left: 0;

top: 0;

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, 0.7);

}

.modal-content {

background-color: #f4f4f4;

margin: 10% auto;

padding: 20px;

border: 1px solid #888;

border-radius: 5px;

text-align: center;

}

.close {

color: #aaa;

float: right;

font-size: 28px;

cursor: pointer;

}

/\* Buttons Styles \*/

button {

padding: 10px 20px;

margin: 10px;

background-color: #4CAF50;

color: #fff;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

button.close-button {

background-color: #f44336;

}

button.close-button:hover {

background-color: #d32f2f;

}

Javascript:

// Get the modal and buttons

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

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

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

const button1 = document.getElementById("button1");

const button2 = document.getElementById("button2");

const button3 = document.getElementById("button3");

// Open the modal

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

modal.style.display = "block";

});

// Close the modal

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

modal.style.display = "none";

});

// Button 1 action

button1.addEventListener("click", () => {

alert("Button 1 clicked");

});

// Button 2 action

button2.addEventListener("click", () => {

alert("Button 2 clicked");

});

// Button 3 action

button3.addEventListener("click", () => {

alert("Button 3 clicked");

});

// Close the modal when clicking outside the modal content

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

if (e.target == modal) {

modal.style.display = "none";

}

});

**Q.6. Use external js library to show slider**

Ans. <!DOCTYPE html>

<html>

<head>

<title>Slick Carousel Example</title>

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

<link rel="stylesheet" type="text/css" href="slick/slick-theme.css">

</head>

<body>

<!-- Slider HTML -->

<div class="slider">

<div><img src="image1.jpg" alt="Image 1"></div>

<div><img src="image2.jpg" alt="Image 2"></div>

<div><img src="image3.jpg" alt="Image 3"></div>

</div>

<!-- Include jQuery and Slick Carousel scripts -->

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<script type="text/javascript" src="slick/slick.min.js"></script>

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

</body>

</html>

$(document).ready(function(){

$('.slider').slick({

dots: true, // Display navigation dots

autoplay: true, // Autoplay the slider

autoplaySpeed: 2000, // Delay between slides in milliseconds

});

});

**Q.7. Prevent the browser when i click the form submit button**

Ans. <!DOCTYPE html>

<html>

<head>

<title>Prevent Form Submission Example</title>

</head>

<body>

<h2>Prevent Form Submission Example</h2>

<form id="myForm">

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

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

<br>

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

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

<br>

<input type="submit" value="Submit">

</form>

<script>

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

form.addEventListener("submit", function(event) {

event.preventDefault(); // Prevent the form submission

// Custom validation or actions can be added here

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

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

if (name === "") {

alert("Name is required");

return;

}

if (email === "") {

alert("Email is required");

return;

}

// If all validation checks pass, you can submit the form programmatically

// form.submit();

});

</script>

</body>

</html>

**New Request**

**Q.1. What is JSON?**

Ans. JSON (JavaScript Object Notation) is a widely used data format that is easy for both humans and computers to work with. It represents data as key-value pairs, similar to objects in many programming languages, and is commonly used for data exchange between systems. JSON is versatile, easy to read, and supported by various programming languages and applications.

**Q.2. What is promises?**

Ans. Promises in JavaScript are a way to handle asynchronous tasks more effectively. They represent an operation that will complete in the future, either successfully (resolved) or with an error (rejected). Promises help make async code more organized and easier to understand.

**Q.3. Write a program of promises and handle that promises also?**

Ans. // A function that returns a promise to simulate an asynchronous task

function simulateAsyncTask(success) {

return new Promise((resolve, reject) => {

setTimeout(() => {

if (success) {

resolve("Task completed successfully.");

} else {

reject("Task encountered an error.");

}

}, 2000);

});

}

// Using the promise and handling it

console.log("Start of the program");

simulateAsyncTask(true)

.then(result => {

console.log("Promise resolved:", result);

return "Additional data from the first .then()";

})

.then(result => {

console.log("Chained .then() with additional data:", result);

return simulateAsyncTask(true); // Simulate another async task

})

.then(result => {

console.log("Second promise resolved:", result);

return simulateAsyncTask(false); // Simulate a failing async task

})

.catch(error => {

console.error("Promise rejected:", error);

return "Fallback data after an error";

})

.then(result => {

console.log("After catching the error:", result);

});

console.log("End of the program");

**Q.4. Use fetch method for calling an api** [**https://fakestoreapi.com/products**](https://fakestoreapi.com/products)**?**

Ans. // Use the fetch method to make an HTTP GET request to the API

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

.then(response => {

if (!response.ok) {

throw new Error("Network response was not ok");

}

return response.json();

})

.then(data => {

// Process the data received from the API

console.log("Data from the API:", data);

})

.catch(error => {

console.error("Error:", error);

});

**Q.5. Display all the product from the api in your HTML page**

Ans. <!DOCTYPE html>

<html>

<head>

<title>Product Listing</title>

</head>

<body>

<h1>Product Listing</h1>

<div id="productList"></div>

<script>

// Function to fetch products from the API and display them

function fetchAndDisplayProducts() {

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

.then(response => {

if (!response.ok) {

throw new Error("Network response was not ok");

}

return response.json();

})

.then(products => {

// Reference to the productList div

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

// Create and append HTML elements for each product

products.forEach(product => {

const productElement = document.createElement("div");

productElement.innerHTML = `

<h3>${product.title}</h3>

<p><strong>Price:</strong> $${product.price}</p>

<p><strong>Category:</strong> ${product.category}</p>

<p>${product.description}</p>

`;

productList.appendChild(productElement);

});

})

.catch(error => {

console.error("Error:", error);

});

}

// Call the function to fetch and display products when the page loads

fetchAndDisplayProducts();

</script>

</body>

</html>

**JavaScript Essentials**

**Q.1. Calculate subtotal price of quantity in JavaScript?**

Ans. // Define the price per unit and quantity

const pricePerUnit = 10; // Replace with your actual price

const quantity = 5; // Replace with the quantity the user has selected

// Calculate the subtotal price

const subtotal = pricePerUnit \* quantity;

// Display the subtotal

console.log(`Subtotal: $${subtotal}`);

**Q.2. What is JavaScript Output method?**

Ans. JavaScript has several methods for producing output:

console.log(): For debugging and development, it logs messages to the browser console.

alert(): Displays a simple dialog box with a message for user notifications.

document.write(): Writes content directly to the HTML document, but it's less common.

Modifying HTML elements: You can change the content of HTML elements for dynamic web content.

**Q.3. How to used JavaScript Output method?**

Ans. To use JavaScript output methods:

console.log(): Write console.log(yourMessage) to log messages to the browser's console during development.

alert(): Write alert(yourMessage) to show a simple pop-up message to users.

document.write(): Use document.write(yourMessage) to directly add content to your webpage (rarely used in modern web development).

Modifying HTML elements: Change an HTML element's content with JavaScript like document.getElementById("elementId").innerHTML = "New content" for dynamic web apps.

**Q.4. How to used JavaScript Events to do all examples?**

Ans. *Display Output with console.log() on Button Click:*

<button id="logButton">Log Message</button>

<script>

document.getElementById("logButton").addEventListener("click", () => {

console.log("Button clicked!");

});

</script>

*Display an alert() on Button Click:*

<button id="alertButton">Show Alert</button>

<script>

document.getElementById("alertButton").addEventListener("click", () => {

alert("This is an alert message.");

});

</script>

*Write to the Document (Not Recommended):*

<button id="writeButton">Write to Document</button>

<script>

document.getElementById("writeButton").addEventListener("click", () => {

document.write("This text will be written to the document.");

});

</script>

*Modify HTML Elements on Button Click:*

<div id="output">Initial content</div>

<button id="changeContentButton">Change Content</button>

<script>

document.getElementById("changeContentButton").addEventListener("click", () => {

document.getElementById("output").innerHTML = "New content";

});

</script>