Exercise 1: JavaScript Debugging

Problem Statement:

You've been given a simple JavaScript code snippet that's intended to toggle the visibility of an element when a button is clicked. However, it's not working as expected.

Code:  
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Toggle Element</title>

</head>

<body>

    <button onclick="toggleElement()">Toggle Element</button>

    <div id="target" style="display: none;">This is the target element.</div>

    <script>

        function toggleElement() {

            var element = document.getElementById("target");

            element.style.display = (element.style.display === "none") ? "block" : "none";

        }

    </script>

</body>

</html>

Tasks:

Identify the issue in the provided JavaScript code.

Debug and fix the code so that clicking the button toggles the visibility of the element.

**Solution -**

The problem in the above-mentioned JavaScript code is that when you click on the button, the element’s style.display property is set to “none”, but the style.display property does not change. Therefore, the original state of the element will always be “none,” and the toggle operation will not work correctly.

To solve this problem, you should keep the current state of your element’s display property and switch it based on the current state. Here is a modified code that solves this problem:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Toggle Element</title>

</head>

<body>

<button onclick="toggleElement()">Toggle Element</button>

<div id="target" style="display: none;">This is the target element.</div>

<script>

function toggleElement() {

var element = document.getElementById("target");

***// The lines below this is Edited***

**if (element.style.display === "none" || element.style.display === "") {**

**element.style.display = "block";**

**} else {**

**element.style.display = "none";**

**}**

**}**

</script>

</body>

</html>

Question -2

Problem Statement:

You've been given an HTML and CSS code snippet that's supposed to create a centered, responsive container. However, it's not displaying as expected.

Solution

Tasks:

Identify the issue in the provided CSS code.

Debug and fix the code so that the container is centered on the page.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Centered Container</title>

*//changes here*

<style>

.container {

width: 50%;

background-color: #f0f0f0;

padding: 20px;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

}

</style>

</head>

<body>

<div class="container">

<h1>Centered Container</h1>

<p>This container is now centered on the page.</p>

</div>

</body>

</html>

These changes will center the container vertically and horizontally on the page. Position: absolute; Top: 50%; Left: 50%; Transform: translate(-50% -50%) Properties work together to center the container.

Exercise 3: Debugging JavaScript Functions

Objective: Identify and fix issues in JavaScript functions.

This code snippet with a JavaScript function that performs a specific task, but contains bugs or inefficiencies.

Debug the function and ensure it works correctly and efficiently.

Code:

function calculateSum(arr) {

let sum = 0;

for (let i = 0; i < arr.length; i++) {

sum += arr[i];

}

return sum;

}

const numbers = [1, 2, 3, 4, 5];

const result = calculateSum(numbers);

console.log(result); // Should output 15

Solution –

The provided code snippet is a simple JavaScript function that calculates the sum of elements in an array. The function appears to be correct. This code should work correctly and efficiently. It calculates the sum of the numbers in the array numbers, and it should output 15 when you run it. There don't appear to be any bugs or inefficiencies in the code.

Question-4

Objective: Identify and fix CSS styling issues to achieve the desired layout.

This code snippet with HTML and CSS code that creates a specific layout, but contains CSS issues like misalignment, overlapping elements, or incorrect colors.

Debug the CSS to achieve the desired layout.

Solution-

*1. The container div has a width of 50%, which means it will take up half of the screen width. However, the container div also has a padding of 20px, which adds extra space around the container. This extra space is included in the 50% width, causing the container to be slightly larger than half of the screen width.*

*To fix this issue, you can change the width of the container div to 100% to ensure it takes up the full width of the screen.*

*2. The box divs are not aligned properly. They are displayed as block-level elements by default, which means they will take up the full width of their parent container. This causes the box divs to be misaligned, as they are pushed down to the next line.*

*To fix this issue, you can change the display property of the box divs to inline-block. This will allow the box divs to be displayed in a horizontal row, with each box div taking up only the necessary width.*

*3. The box divs have a background color of #007bff, but the text color is set to #ffffff. This makes it difficult to read the text inside the box divs.*

To fix this issue, you can change the text color to a darker color, such as #333333.

Changes Made to the Code

.container {

 width: 100%;

 margin: 0 auto;

 background-color: #f0f0f0;

 padding: 20px;

 text-align:center;

}

.box {

 width: 100px;

 height: 100px;

 background-color: #007bff;

 color: #333333;

 text-align: center;

 line-height: 100px;

 display: inline-block;

}