**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**

<!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");

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

element.style.display = "block";

} else {

element.style.display = "none";

}

}

</script>

</body>

</html>

The potential problem with the provided JavaScript code arises when it relies on the `element.style.display` property. This code may work as intended only when the initial state of the element is explicitly set to "none" within the inline style. When the inline style is not defined, the `element.style.display` property will be an empty string by default. The given solution addresses this issue by checking for both an empty string and the "none" value to properly toggle the `display` property.

**Exercise 2: CSS Troubleshooting**

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.

Code:

<!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>

<style>

.container {

margin: auto;

width: 50%;

background-color: #f0f0f0;

padding: 20px;

}

</style>

</head>

<body>

<div class="container">

<h1>Centered Container</h1>

<p>This container should be centered on the page.</p>

</div>

</body>

</html>

Tasks:

Identify the issue in the provided CSS code.

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

**SOLUTION**

<!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>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.container {

width:50%;

background-color: #f0f0f0;

padding: 20px;

text-align:center;

}

</style>

</head>

<body>

<div class="container">

<h1>Centered Container</h1>

<p>This container is centered both horizontally and vertically on the page.</p>

</div>

</body>

</html>

In the provided code, the container was initially centered horizontally but not vertically. To address this, the following adjustments were made:

1. Instead of styling the container directly, the `body` element was utilized as the flex container by applying `display: flex;`.

2. The `justify-content: center` and `align-items: center` properties were applied to the `body` element. This combination ensured that the contents were centered both horizontally and vertically within the viewport.

3. The `height: 100vh;` property was set on the `body` element. This modification guarantees that the `body` element takes up the full height of the viewport, enabling the container to be centered vertically.

**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**

<html>

<head>

<script>

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

</script>

</head>

</html>

In the given code, the <html> <head> and <script> tags were missing. By adding the tags in the above code, we get 15 as the output.

**Exercise 2: Debugging CSS Styling Issues**

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.

Code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Styling Debugging Exercise</title>

<style>

.container {

width: 50%;

margin: 0 auto;

background-color: #f0f0f0;

padding: 20px;

}

.box {

width: 100px;

height: 100px;

background-color: #007bff;

color: #ffffff;

text-align: center;

line-height: 100px;

}

</style>

</head>

<body>

<div class="container">

<div class="box">Box 1</div>

<div class="box">Box 2</div>

<div class="box">Box 3</div>

</div>

</body>

</html>

**SOLUTION**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Styling Debugging Exercise</title>

<style>

.container {

width: 100%;

text-align: center; /\* Center the content horizontally \*/

}

.box {

display: inline-block; /\* Display boxes inline \*/

width: 100px;

height: 100px;

background-color: #007bff;

color: #ffffff;

text-align: center;

line-height: 100px;

margin: 10px; /\* Add some margin between boxes \*/

}

</style>

</head>

<body>

<div class="container">

<div class="box">Box 1</div>

<div class="box">Box 2</div>

<div class="box">Box 3</div>

</div>

</body>

</html>

Modifications were applied as follows:

1. Within the `.container` class, the `width` property was adjusted to occupy the full width of its parent container. Additionally, `text-align: center` was employed to horizontally center the content within the container.

2. For the `.box` class, the `display` property was set to `inline-block` to ensure that the boxes are arranged horizontally. A margin was also introduced to create some spacing between the individual boxes.