**EX-1:**

<!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>

Explanation:

The JavaScript code you provided is close to working, but there's a small issue. When you use **element.style.display**, it returns an empty string by default, not **"none"** as you're checking for. To fix the code, you should first check if the element is currently displayed as "none" and then toggle it to "block," or vice versa. In this corrected code, it first checks if the element's display property is "none" or an empty string (initial state), and if so, it sets it to "block." Otherwise, if it's not "none," it sets it to "none," effectively toggling the visibility of the element when the button is clicked.

**EX-2:**

<!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 {

  position: absolute;

  top: 50%;

  left: 50%;

  transform: translate(-50%, -50%);

  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>

Explanation:

The CSS code you provided appears to be correct for centering the container horizontally. However, there is a missing CSS property to center the container vertically. In this updated code, I've added the **position: absolute;** property to allow for centering both horizontally and vertically. The **top: 50%;** and **left: 50%;** properties move the container to the center of the page, and the **transform: translate(-50%, -50%);** property then centers it perfectly.

**EX-3:**

EX-1:

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

**no errors**

Explanation:

There are no apparent bugs or inefficiencies in this code. It's a straightforward and efficient way to calculate the sum of elements in an array. If you are experiencing issues with it, please double-check that there are no other parts of your code interfering with its operation, and ensure that the **numbers** array is defined correctly with the expected values.

Ex-2:

<!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;

display: flex; /\* Add display: flex; to create a flex container \*/

justify-content: space-between; /\* Add space between the boxes \*/

}

.box {

width: 100px;

height: 100px;

background-color: #007bff;

color: #ffffff;

text-align: center;

line-height: 100px;

margin: 5px; /\* Add margin to separate the 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>

Explanation:

Changes made:

1. Added **display: flex;** to the **.container** to make it a flex container.
2. Added **justify-content: space-between;** to the **.container** to evenly space the boxes within the container.
3. Added **margin: 5px;** to the **.box** to separate the boxes and fix overlapping issues.

With these changes, the boxes will be evenly spaced and aligned within the container.