VaultofCodes Internship – Week 2 Debugging Exercise

Name: Adrian Mathew Aloysius

Domain: Web Development

Institute: New Horizon College of Engineering

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.

Incorrect 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.

Correct 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">This is the target element.</div>
    <script>
        function toggleElement() {
            var element = document.getElementById("target");
            if (element.style.visibility === 'hidden') {
            element.style.visibility = 'visible';
            } else {
            element.style.visibility = 'hidden';
            }
        }
    </script>
</body>
</html>
```

Explanation:

The issue with the original JavaScript code is that it tries to directly toggle the display property of the element without checking its initial state. The code assumes that the element is initially hidden (style.display set to "none"), and then it tries to toggle it between "block" and "none." If the element's initial state is different, this code won't work as expected. It also uses the ‘display:none’ and ‘display:block’ property to toggle elements on and off. Using the display property removes the element from the layout/document without preserving its position.

In the updated code we use the visibility property to control the visibility of the element. By using ‘visibility:hidden’ and ‘visibility:visible’ property we can still preserve the space occupied by the target element when it is hidden.

We check if the element's current display style is "hidden". If it is, we set it to "block" to make it visible. If it's not "none," we set it to "none" to hide it. This approach should work regardless of the initial state of the element.

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.

Incorrect Code:

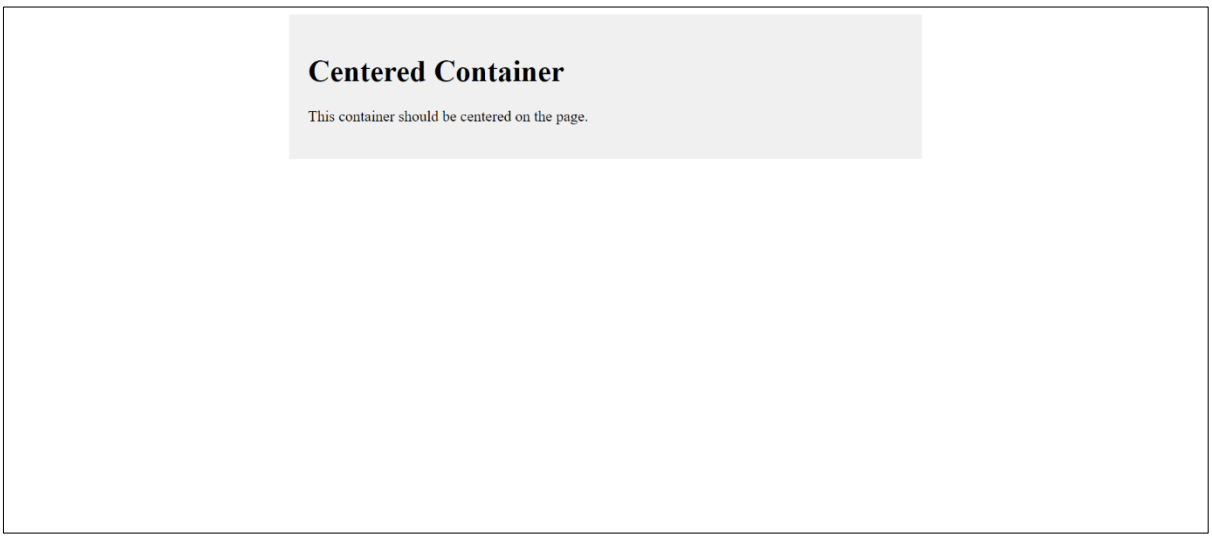
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Centered Container</title>
<style>
.container {
margin: auto;
width: 50%;
background-color: #f0f0f0;
padding: 20px;
}
</style>
</head>
<body>
<div class="container">
<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 cantered on the page.

Incorrect Output:



Correct 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 {
position: absolute;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
text-align: center;
width: 50%;
background-color: #f0f0f0;
padding: 20px;
}
</style>
</head>
<body>
<div class="container">
```

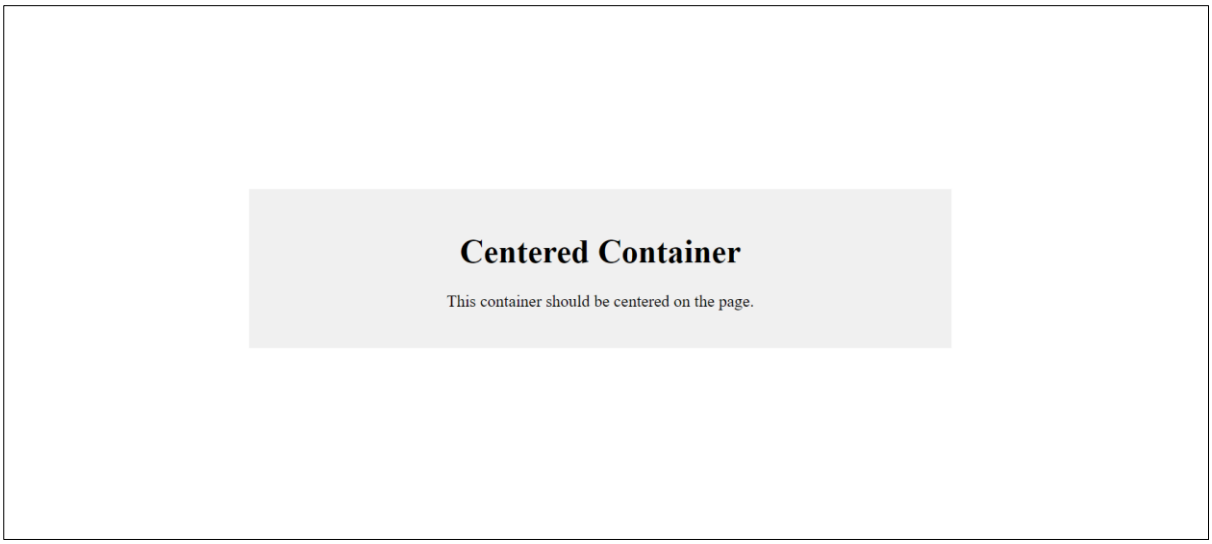
<h1>Centered Container</h1>

This container should be centered on the page.

</body>

</html>

Correct Output:



Explanation:

The original code only centres the container in the centre horizontally but not vertically. In the updated code, we set the position of the element to absolute to position the container relative to the viewport. top:50% and left:50% move it to the centre horizontally and vertically. transform: translate(-50%, -50%) is used to fine-tune the centering. The text within the container is also aligned in the centre of the container box.

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.

Inefficient 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
```

Efficient Code:

```
function calculateSum(arr) {
    return arr.reduce((accumulator, current) =>  accumulator + current , 0);
}
const numbers = [1, 2, 3, 4, 5];
const result = calculateSum(numbers);
console.log(result); //Should Output 15
```

Explanation:

The JavaScript function calculateSum appears to be correct and efficient. It is designed to calculate the sum of the elements in an array, and it works as expected but using the for-loop is not the most efficient way.

In the updated code we use a reduce function that reduces all elements in the array to a single value and it has 2 arguments. A callback function as the first argument and the initial value of the accumulator variable. The accumulator variable stores the last returned value from the previous function call. The current variable represents the current array item. The second argument represents the initial value of the accumulator which is 0 in this case.

Exercise 4: 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.

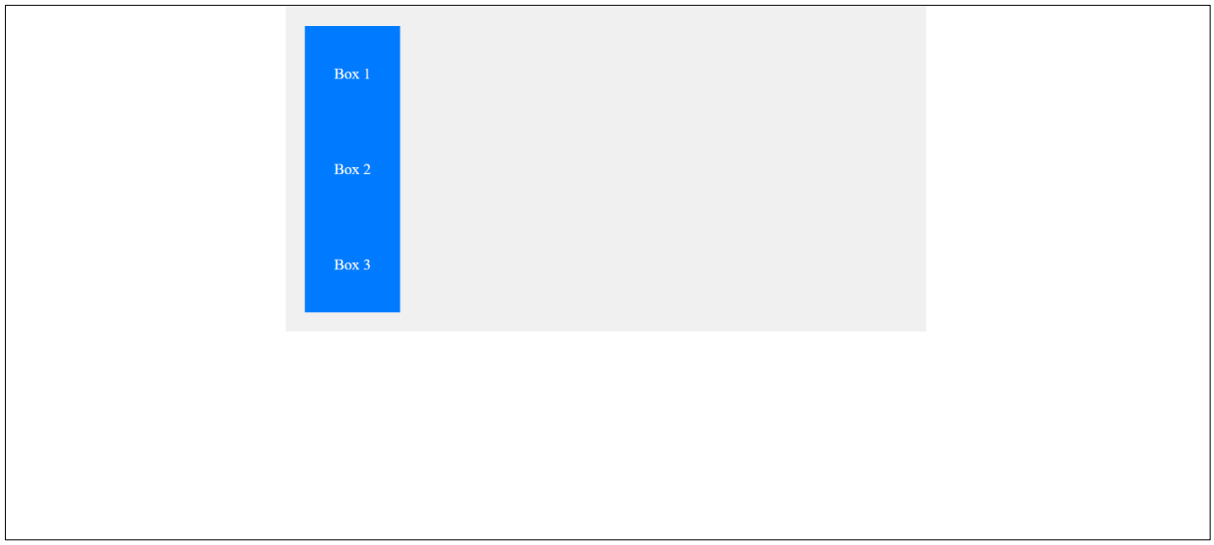
Debug the CSS to achieve the desired layout.

Incorrect 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>
```

Incorrect Output:

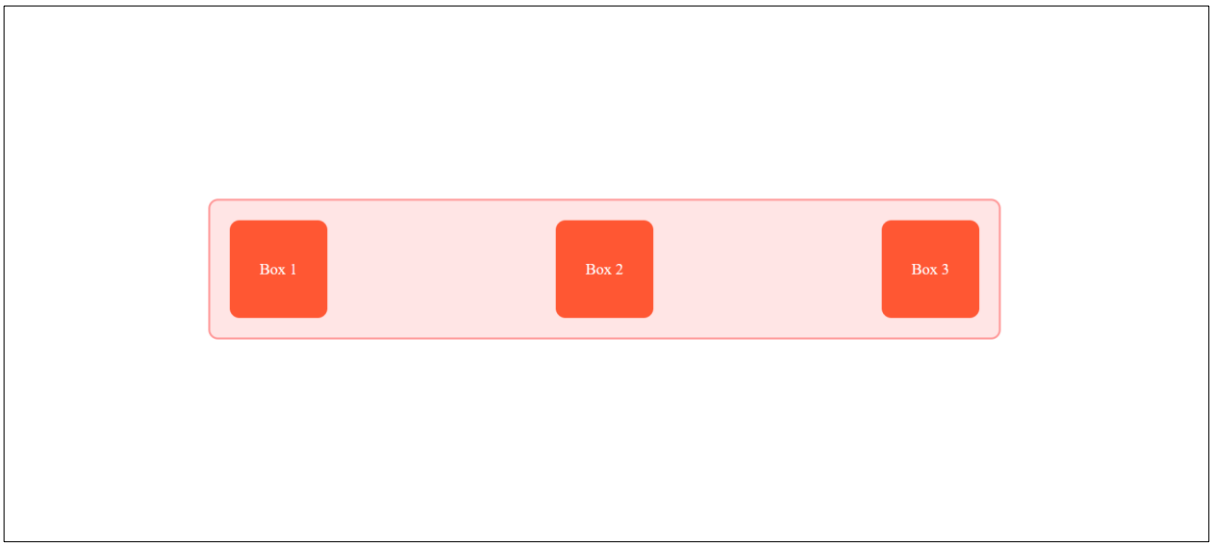


Correct Code:

```
<!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>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
```

```
        .container {
            display: flex;
            justify-content: space-between;
            align-items: center;
            width: 60%;
            background-color: #ffe5e5;
            padding: 20px;
            border: 2px solid #ff9999;
            border-radius: 10px;
        }
    .box {
            width: 100px;
            height: 100px;
            background-color: #ff5733;
            color: #ffffff;
            text-align: center;
            line-height: 100px;
            border-radius: 10px;
        }
    </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>
```

Correct Output:



Explanation:

The original code contains misalignment issues, overlapping elements and incorrect colours.

In the updated code here are the modifications made:

- Centered the container horizontally using ‘display:flex’ and vertically within the viewport.
- Added a border to the container with a border radius for a rounded appearance.
- Adjusted the container's width to 60%.
- Added spacing and alignment properties to improve element positioning.
- Increased the border radius for the boxes for rounded corners.
- Improved the overall color scheme by using a light pink background for the container, a lighter pink border, and a coral color for the boxes for better readability and aesthetics.