DATE : 06/08/2024

Part 1: CSS Positioning

Objective: Create a web page demonstrating different CSS positioning techniques.

Instructions:

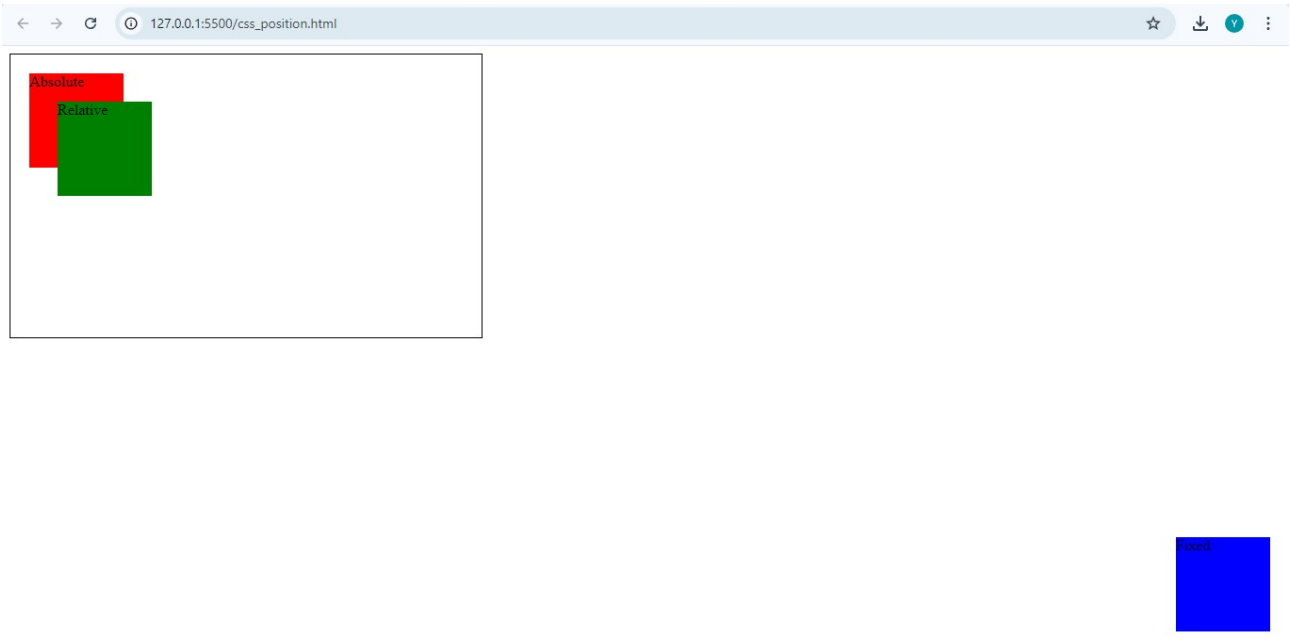
1. Create an HTML file named index.html.
2. Add a div element with the class container and three child div elements with classes absolute, relative, and fixed.
3. Style the container to have a width of 500px and height of 300px.
4. Apply different positioning styles to each child div.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>CSS Positioning</title>
   <style>
            .container {
                width: 500px;
                height: 300px;
                 border: 1px solid black;
                  position: relative;
              }
             .absolute {
                    position: absolute;
                     top: 20px;
                     left: 20px;
                     width: 100px;
                    height: 100px;
                     background-color: red;
            }
            .relative {
                     position: relative;
                     top: 50px;
                    left: 50px;
                    width: 100px;
                     height: 100px;
                     background-color: green;
             }
            .fixed {
                     position: fixed;
                     bottom: 20px;
```

```
            right: 20px;
            width: 100px;
            height: 100px;
            background-color: blue;
        }
    </style>
</head>
<body>
        <div class="container">
            <div class="absolute">Absolute</div>
            <div class="relative">Relative</div>
            <div class="fixed">Fixed</div>
        </div>
</body>
</html>
```

Output:



Part 2: Try changing the width and give only 10px to border property. Mention what changes you have noticed with the content. Hint: Create a html with div containers and classes accordingly.

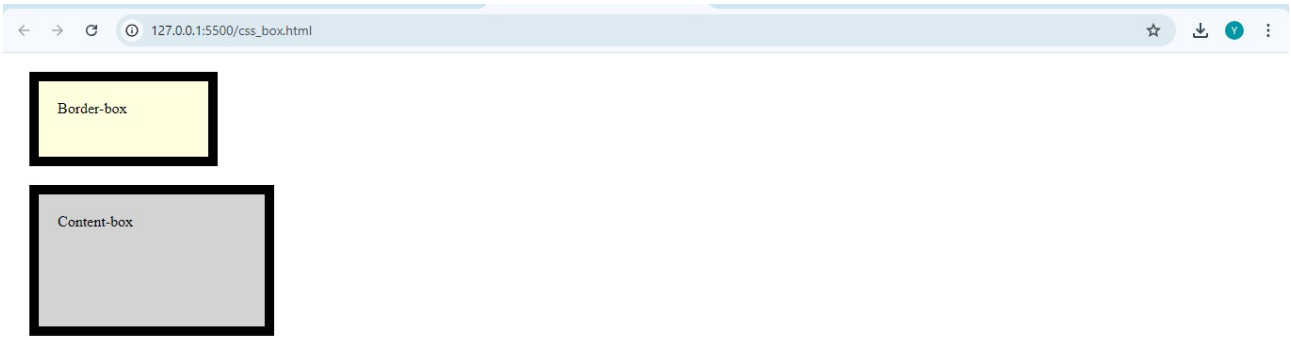
```
.border-box, .content-box {
    width: 200px;
    height: 100px;
    margin: 20px;
    padding: 20px;
```

```
    border: 10px solid black;
}
.border-box {
    box-sizing: border-box;
    background-color: lightyellow;
}
.content-box {
    box-sizing: content-box;
    background-color: lightgray;
}
```

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Box Sizing</title>
    <style>
        .border-box, .content-box {
            width: 200px;
            height: 100px;
            margin: 20px;
            padding: 20px;
            border: 10px solid black;
        }
        .border-box {
            box-sizing: border-box;
            background-color: lightyellow;
        }
        .content-box {
            box-sizing: content-box;
            background-color: lightgray;
        }
    </style>
</head>
<body>
    <div class="border-box">Border-box</div>
    <div class="content-box">Content-box</div>
</body>
</html>
```

Output:



Part 3: Javascript – show difference between substr and substring with negative index and positive index for the string “The world is wonderful”.

Code:

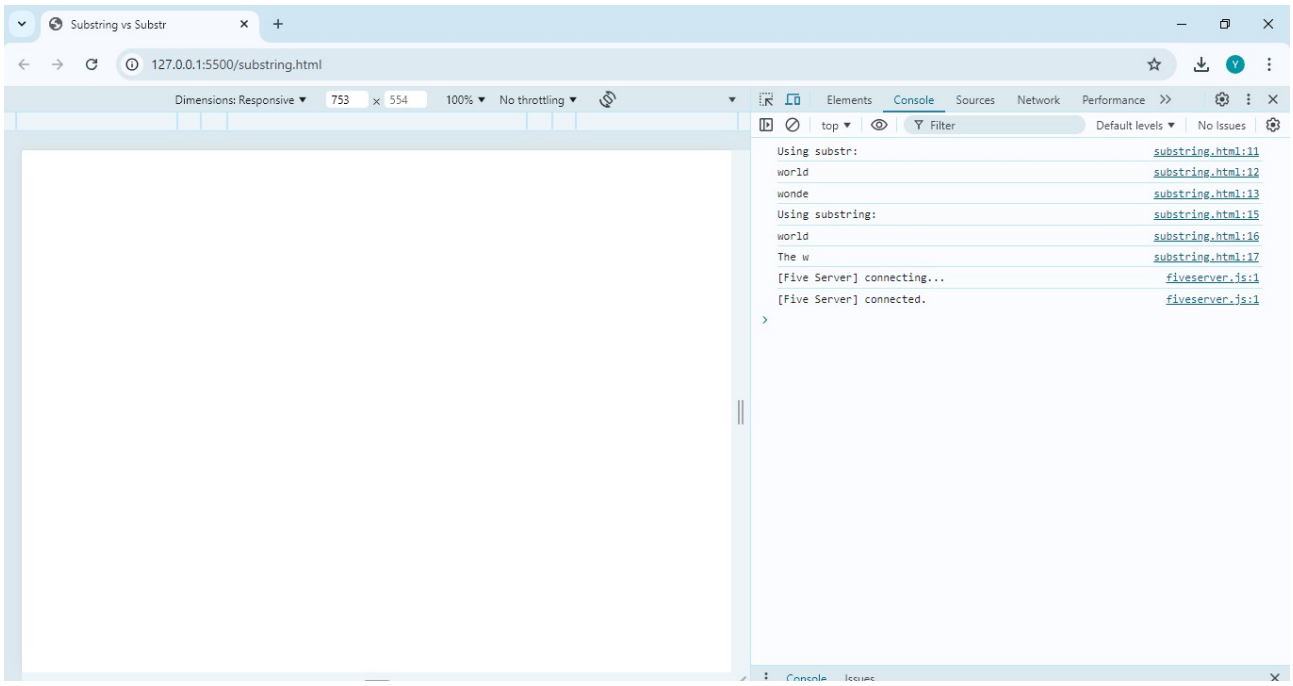
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Substring vs Substr</title>
  <script>
    window.onload = function() {
      var str = "The world is wonderful";

      console.log("Using substr:");
      console.log(str.substr(4, 5)); // world
      console.log(str.substr(-9, 5)); // wonde

      console.log("Using substring:");
      console.log(str.substring(4, 9)); // world
      console.log(str.substring(-9, 5)); // The w (negative index treated as 0)
    }
  </script>
```

```
</head>
<body>
</body>
</html>
```

Output:

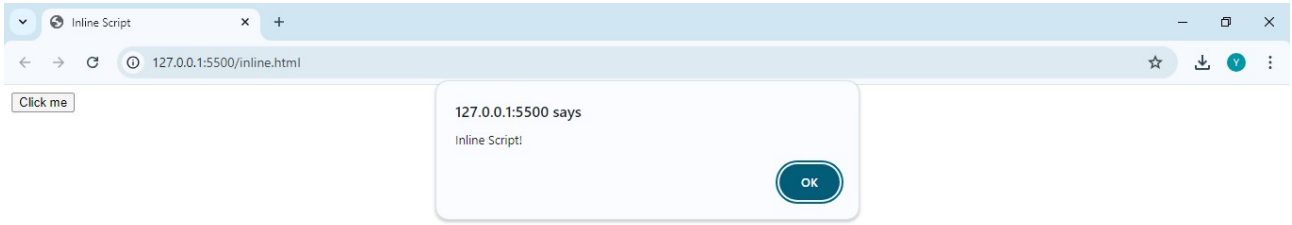


Part 4: Javascript : Show what's inline, internal and external scripts.

Inline Script:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Inline Script</title>
</head>
<body>
  <button onclick="alert('Inline Script!')">Click me</button>
</body>
</html>
```

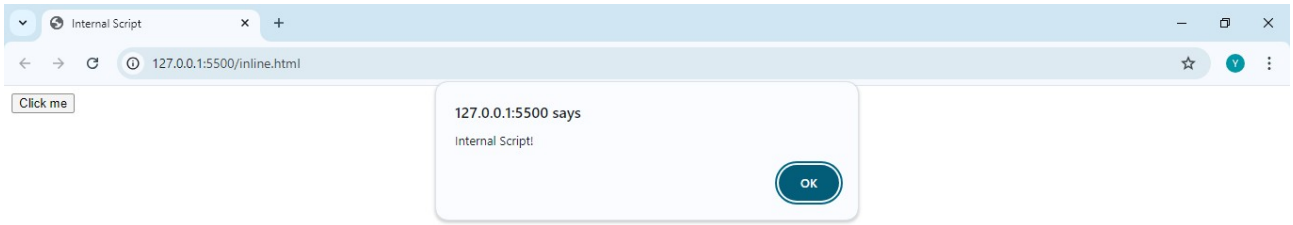
Output:



Internal Script:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Internal Script</title>
   <script>
      function showAlert() {
         alert('Internal Script!');
      }
   </script>
</head>
<body>
   <button onclick="showAlert()">Click me</button>
</body>
</html>
```

Output:



External Script:

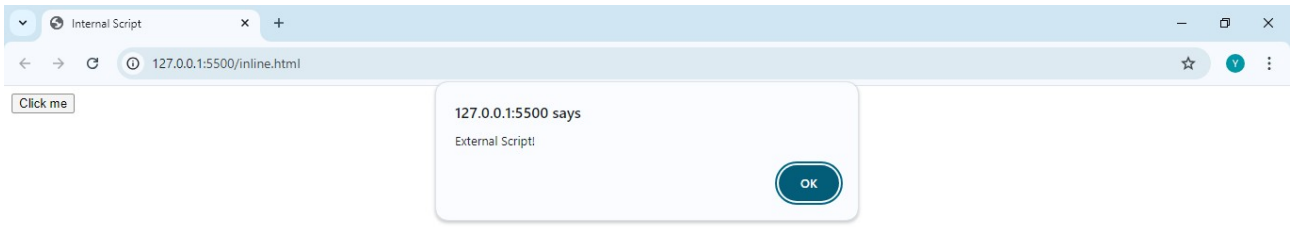
index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>External Script</title>
   <script src="script.js"></script>
</head>
<body>
   <button onclick="showAlert()">Click me</button>
</body>
</html>
```

script.js:

```
function showAlert() {
        alert('External Script!');
}
```

Output:



Part 5: Javascript: As per naming convention, which variable is advisable to use for functions or arrays: const or let or var?

Explanation:

- **const:** Use for variables that are not going to be reassigned. Best for constants and also for arrays and objects that you don't want to reassign.
- **let:** Use for variables that may change their value but are block-scoped. Preferred for loops, conditionals, and when reassignment is necessary within a block scope.
- **var:** Use for variables that are function-scoped. It has been largely superseded by let and const and is generally not recommended in modern JavaScript due to potential scope issues.

Best Practice:

- Use const for functions and arrays whenever possible.
- Use let when you need a variable to be reassigned within a block scope.
- Avoid using var.

Example:

```
// Using const for a function
const myFunction = function() {
  console.log('This is a function');
```

```
}
// Using const for an array
const myArray = [1, 2, 3];
// Reassigning elements of the array is allowed
myArray.push(4);
// Using let for a variable that changes
let counter = 0;
counter++;
```