9 – DAY – TASK (06-08-2024)

Part 1: CSS Positioning

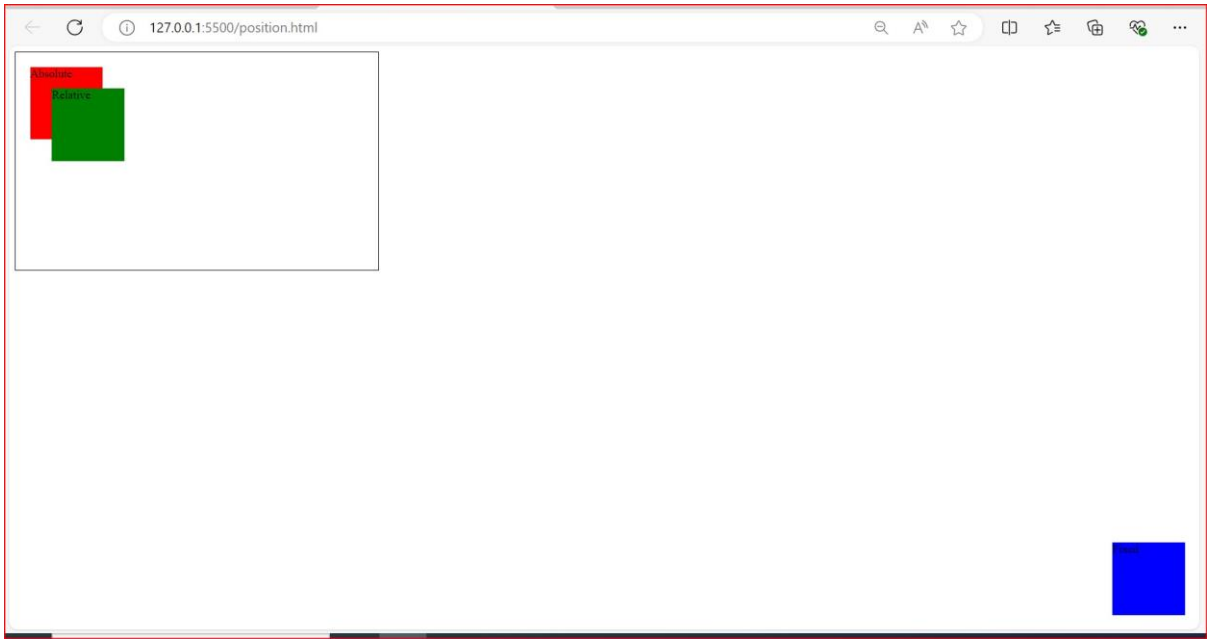
1. Create a web page demonstrating different CSS positioning techniques.
   Instructions:
   1. Create an HTML file named index.html.
   2. Add a div element with the class container and three child div elements with classes absolute, relative, and fixed.
   3. Style the container to have a width of 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;

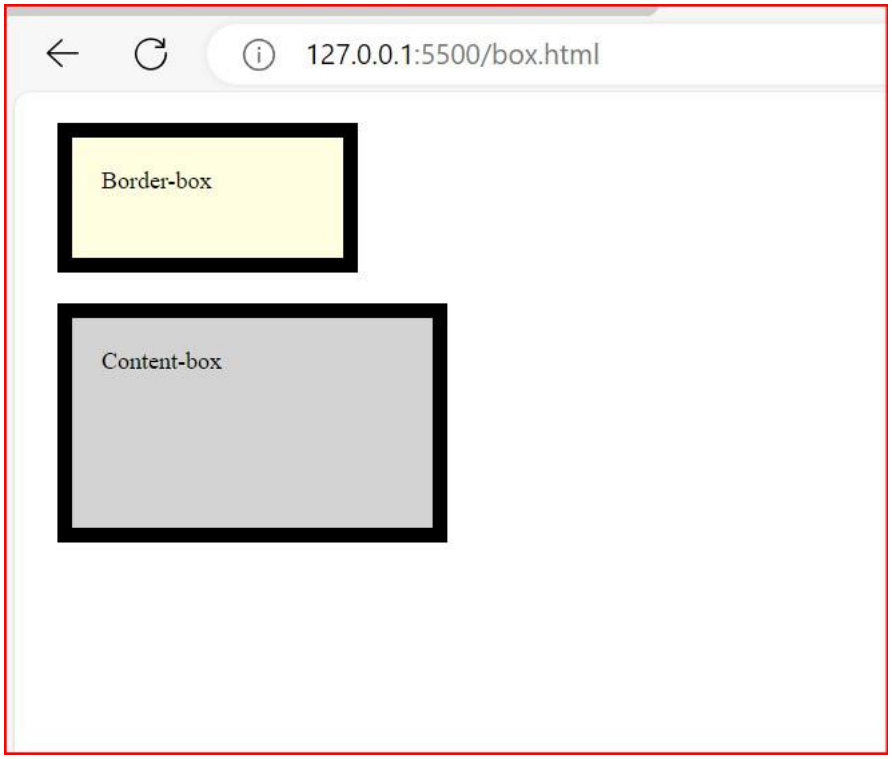
```
  margin: 20px;
  padding: 20px;
  border: 10px solid black;
}
.border-box {
  box-sizing: border-box;
  background-color: lightyellow;
}
.content-box {
  box-sizing: content-box;
  background-color: lightgray;
}
```

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:



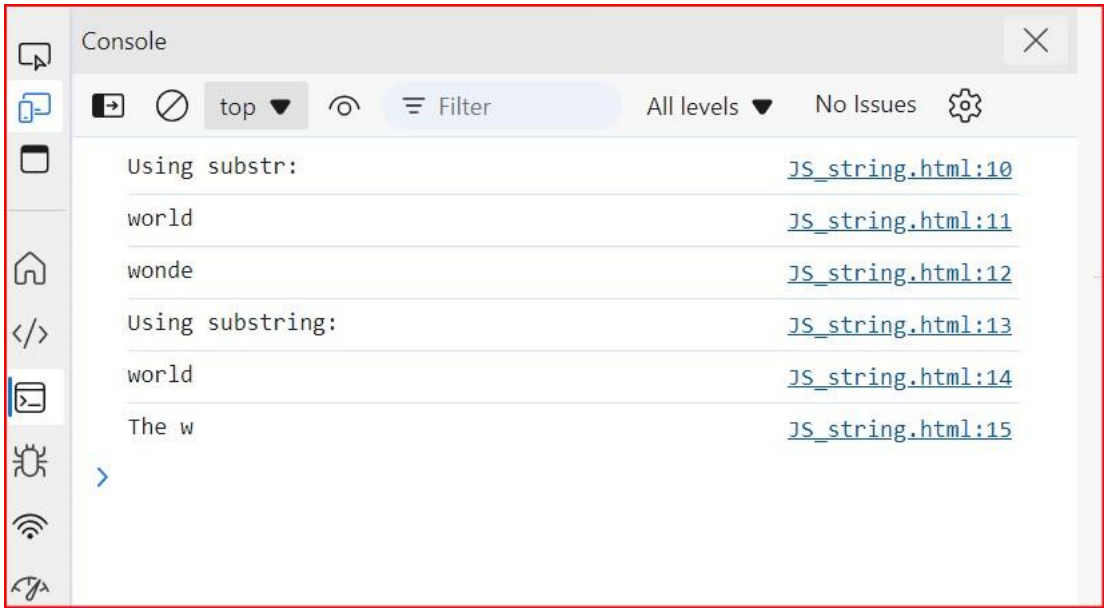
Part 3: Javascript – show difference between substr and substring with negative index and positive index for the string “The world is wonderful”.

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>
</html>
```

Output:



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

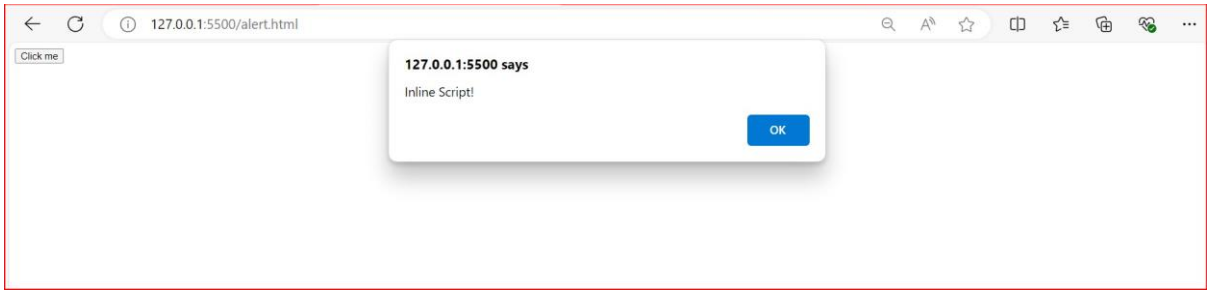
Inline Script:

Code:

```
<!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:

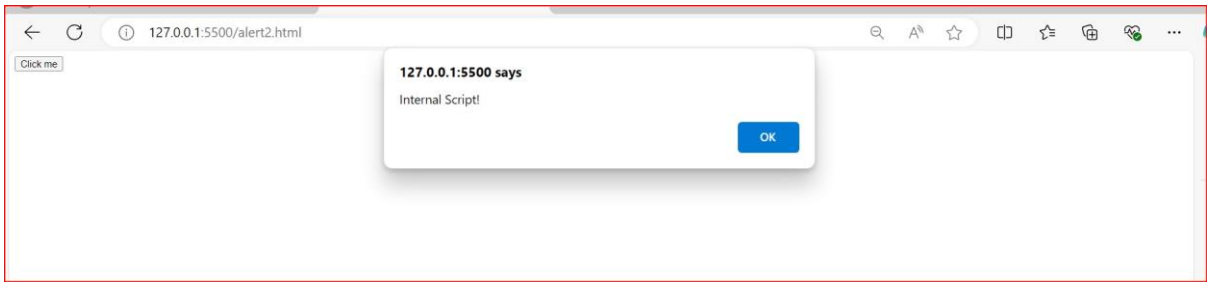


Internal Script:

Code:

```
<!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:

Code:

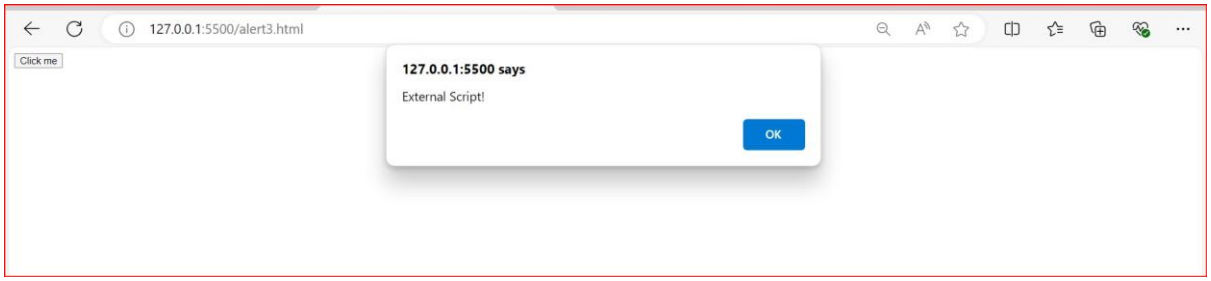
Html file:

```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>
```

JavaScript file:

```javascript
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?

`const`:

- **Functions**: Use `const` to declare functions if you don't need to reassign the function itself. This makes it clear that the reference to the function will not change.

const add = (a, b) => a + b;

- **Arrays**: Use `const` for arrays if you don't need to reassign the array itself, though you can still modify the contents of the array.

```
const numbers = [1, 2, 3];
numbers.push(4); // Allowed
numbers = [1, 2, 3, 4]; // Error: Assignment to constant variable.
```

`let`:

- **Functions**: Generally, `let` is not used for functions since `const` is preferable for function declarations to prevent reassignment. However, `let` can be used within blocks if a function needs to be redefined within a specific scope.
- **Arrays**: Use `let` if you need to reassign the entire array variable.

```
let numbers = [1, 2, 3];

numbers = [4, 5, 6]; // Allowed
```

`var`:

- **Functions**: `var` can be used for functions, but it has function scope (or global scope) and can lead to confusing bugs because it is hoisted. It's generally better to use `const` or `let` for functions.
- **Arrays**: Similar to functions, using `var` for arrays is less common in modern code. `var` has function scope and can be error-prone due to hoisting. Prefer `const` or `let` based on whether reassignment is needed.

```
function example() {

 if (true) {var message = "Hello, World!";}

console.log(message); // "Hello, World!"}

example();


function hoistingExample() { console.log(x); // undefined

var x = 10;

console.log(x); // 10

}

hoistingExample();


var globalVar = "I'm a global variable";

function testVar() {
```

```
var localVar = "I'm a local variable";
console.log(globalVar); // "I'm a global variable"
console.log(localVar);  // "I'm a local variable"
}

testVar();
console.log(globalVar); // "I'm a global variable"
console.log(localVar);  // ReferenceError: localVar is not defined
```