

LAB ASSIGNMENT NO – 5a

Aim: Write a Javascript Program to study arrow functions, DOM Manipulation and CSS Manipulations

Theory:

1. Arrow Functions:

It refers to anonymous functions in programming.

- Lambda functions are a concise mechanism to represent anonymous functions.
- These functions are also called as Arrow functions.
- There are 3 parts to a Lambda function –
- Parameters– A function may optionally have parameters.
- The fat arrow notation/lambda notation(=>): It is also called as the goes to operator.
- Statements– Represents the function's instruction set.
([param1, param2,...param n])=>statement
- Arrow functions remove the need to type out the keyword function every time you need to create a function.
- Instead, you first include the parameters inside the () and then add an arrow => that points to the function body surrounded in { }.

2. DOM Manipulation:

DOM manipulation in JavaScript refers to the process of programmatically interacting with the Document Object Model (DOM) of an HTML or XML document. The DOM represents the structure of the document and allows JavaScript to access and modify its elements, attributes, and content dynamically.

It is used for:

Accessing Elements: You can use JavaScript to access elements in the DOM using methods like getElementById, getElementsByClassName, querySelector, and more.

Modifying Content: JavaScript can change the content of HTML elements by manipulating their textContent, innerHTML, or value properties. This allows you to update text, add or remove elements, and more.

Modifying Attributes: You can modify HTML element attributes such as src, href, class, id, and more using JavaScript. This is often used for dynamic data binding and interaction.

Adding and Removing Elements: JavaScript can create new elements and append them to the DOM, or remove existing elements. This is useful for building dynamic user interfaces.

Styling Elements: You can change the CSS styles of elements using JavaScript by modifying their style properties. This allows you to create interactive and visually appealing web applications.

3. CSS Manipulation:

CSS manipulation in JavaScript involves dynamically changing the styles of HTML elements on a web page using JavaScript code. This manipulation allows you to alter the appearance and layout of elements in response to user interactions, events, or other programmatic conditions.

It can be used for:

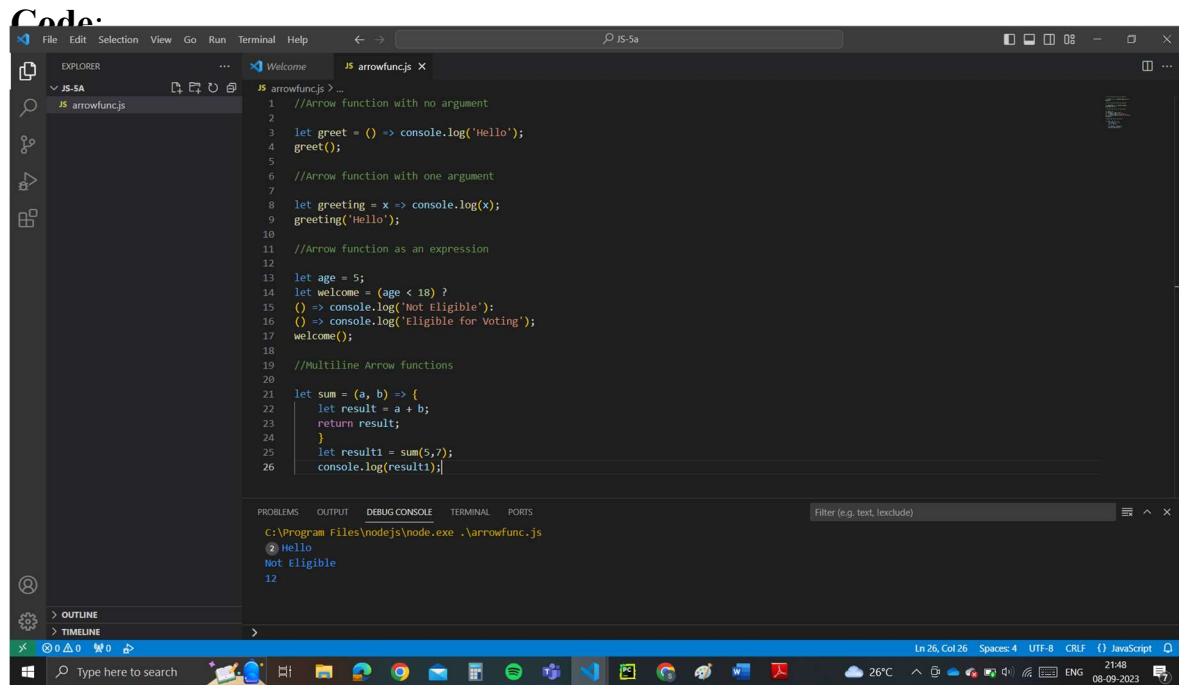
Accessing CSS Styles: JavaScript can access and modify CSS styles applied to HTML elements. You can retrieve the styles of an element using properties like `element.style` or `window.getComputedStyle(element)`.

Modifying CSS Properties: JavaScript can change individual CSS properties of an element by assigning new values to them. For example, you can change an element's color, font-size, width, height, background-color, and more.

Adding and Removing Classes: JavaScript can add or remove CSS classes from elements using the `classList` property. This is a common technique for applying predefined styles to elements or toggling styles based on user actions.

Inline Styles: You can apply inline styles directly to an element using JavaScript. This involves setting the `style` attribute of an element with specific CSS property-value pairs.

Conclusion: This experiment provided valuable insights into the use of modern JavaScript features, including arrow functions, for streamlined event handling. Additionally, the exploration of DOM manipulation techniques allowed for the dynamic creation and modification of HTML elements, enhancing interactivity. The experiment also delved into the realm of CSS manipulation, showcasing the ability to dynamically style elements, providing responsive and visually appealing web experiences.

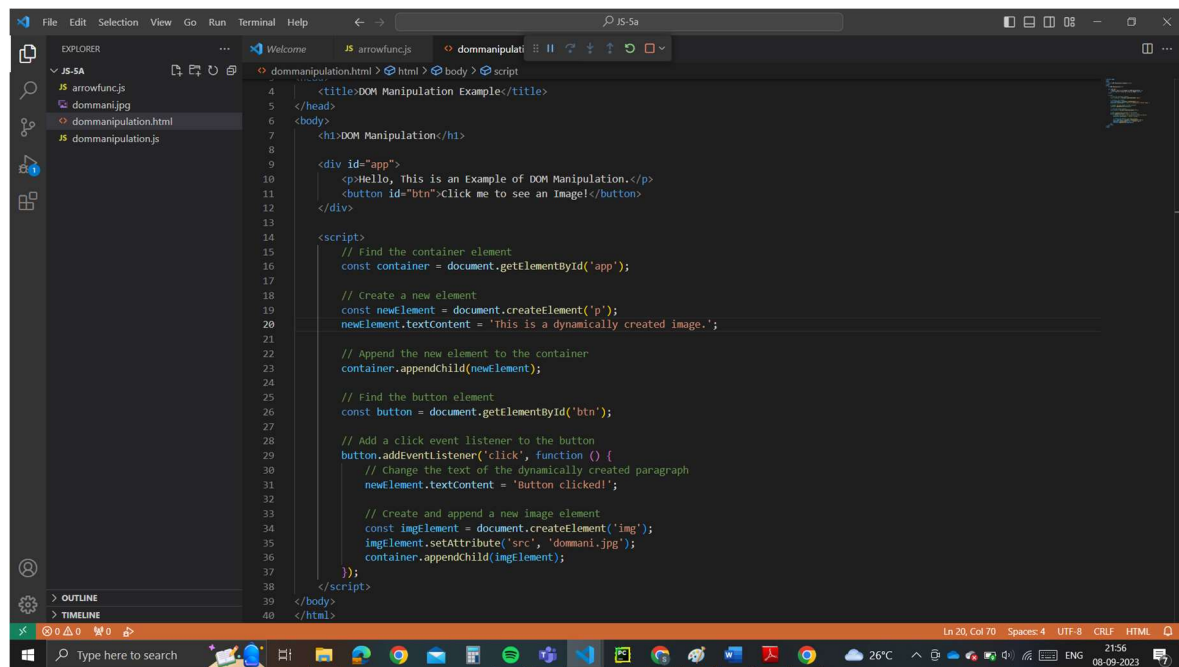


The screenshot shows the VS Code editor with a file named `arrowfunc.js` open. The code defines several arrow functions: a no-argument function `greet`, a one-argument function `greeting`, a function `welcome` using a ternary operator, and a multiline arrow function `sum`. The terminal at the bottom shows the output of running the code: `Hello`, `Not Eligible`, and `12`.

```
1 //Arrow function with no argument
2
3 let greet = () => console.log('Hello');
4 greet();
5
6 //Arrow function with one argument
7
8 let greeting = x => console.log(x);
9 greeting('Hello');
10
11 //Arrow function as an expression
12
13 let age = 5;
14 let welcome = (age < 18) ?
15   () => console.log('Not Eligible');
16   () => console.log('Eligible for Voting');
17   welcome();
18
19 //Multiline Arrow functions
20
21 let sum = (a, b) => {
22   let result = a + b;
23   return result;
24 }
25 let result1 = sum(5,7);
26 console.log(result1);
```

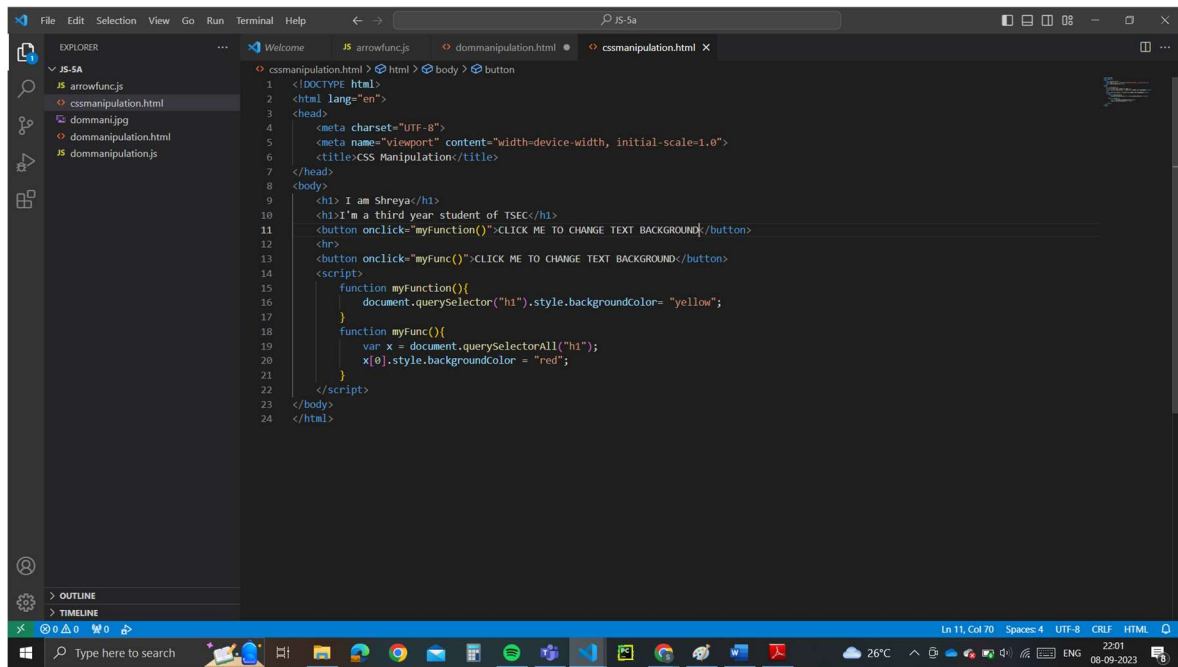
Terminal Output:

```
2 Hello
3 Not Eligible
4 12
```

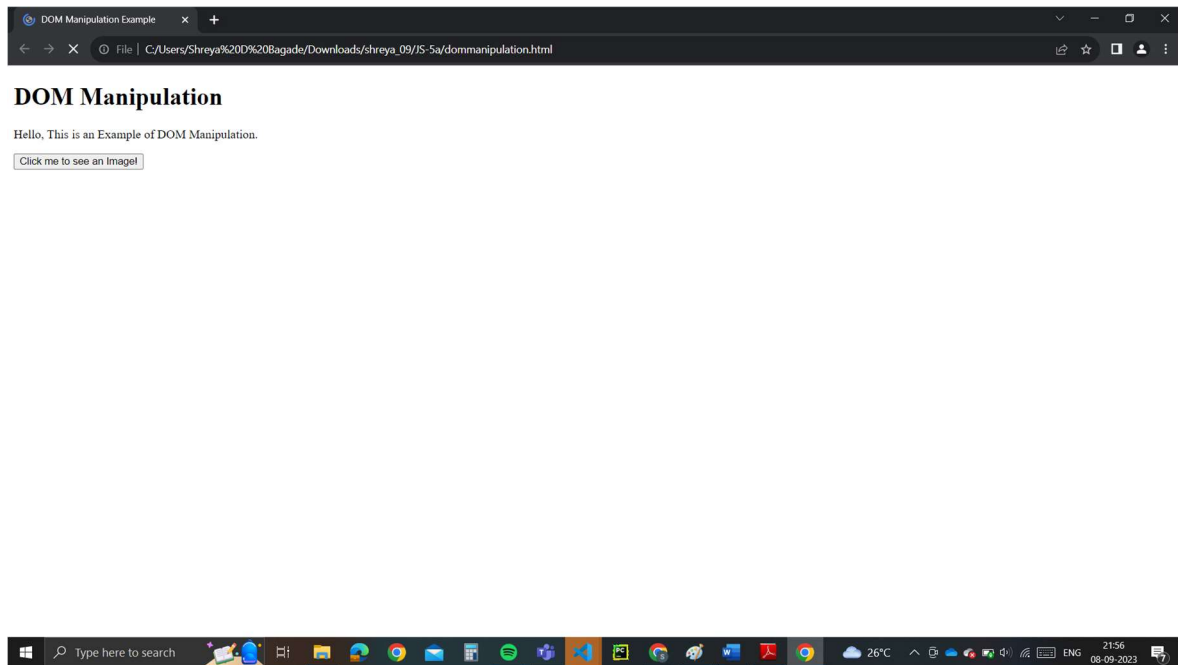


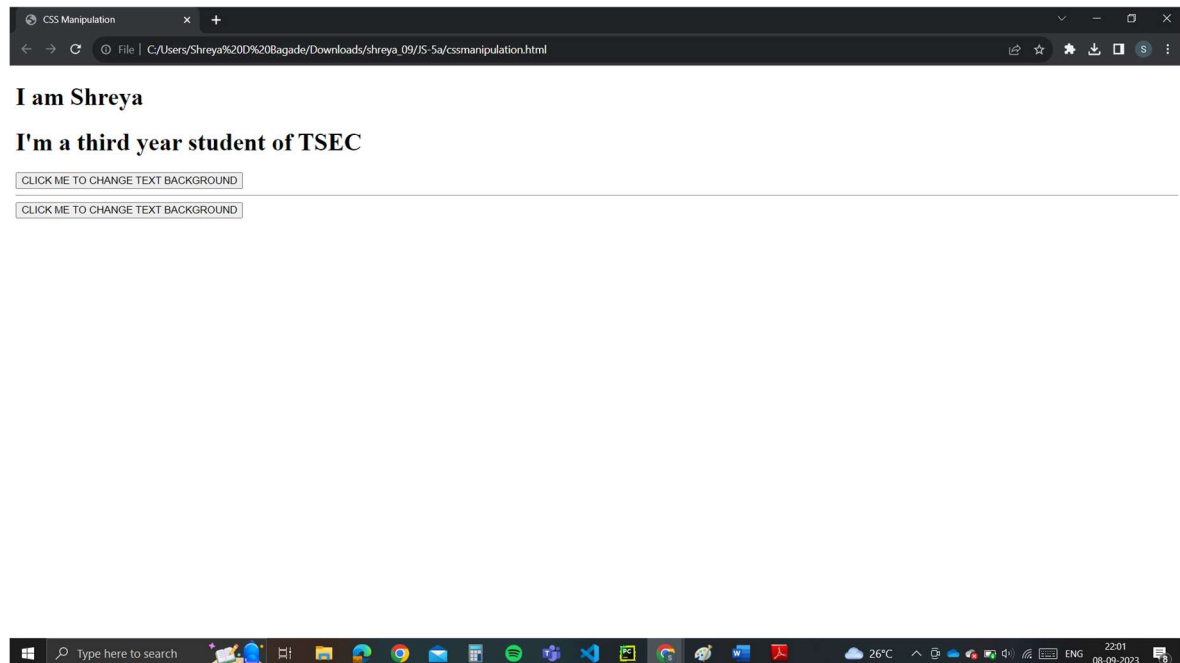
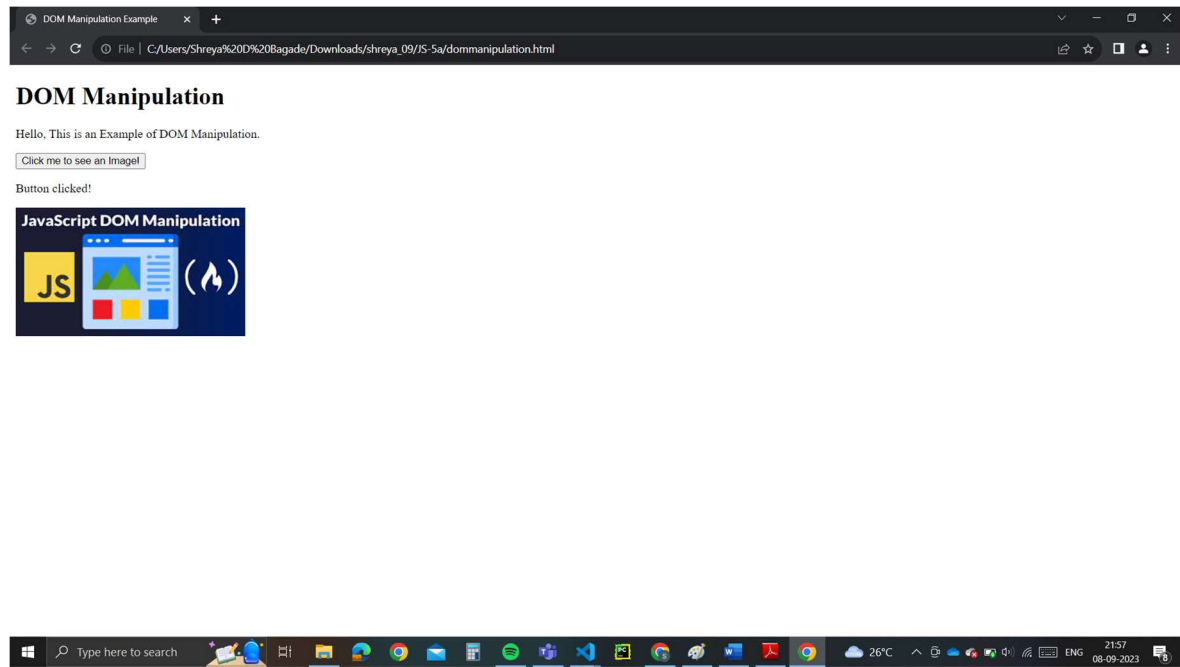
The screenshot shows the VS Code editor with a file named `dommanipulation.js` open. The code demonstrates DOM manipulation by finding a container element, creating a new paragraph element, appending it to the container, finding a button element, adding a click event listener to the button, and creating and appending a new image element.

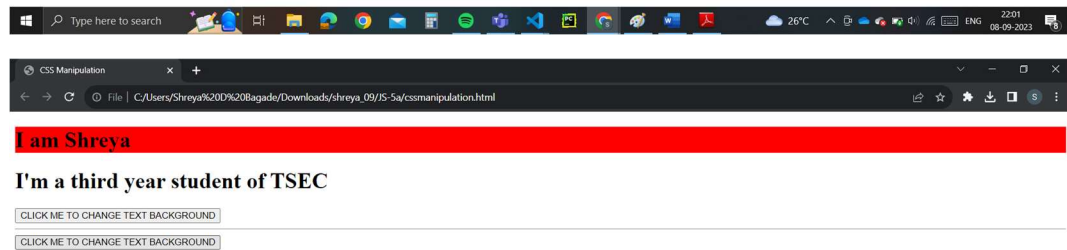
```
4 <title>DOM Manipulation Example</title>
5 </head>
6 <body>
7   <h1>DOM Manipulation</h1>
8
9   <div id="app">
10     <p>Hello, This is an Example of DOM Manipulation.</p>
11     <button id="btn">Click me to see an Image!</button>
12   </div>
13
14   <script>
15     // Find the container element
16     const container = document.getElementById('app');
17
18     // Create a new element
19     const newElement = document.createElement('p');
20     newElement.textContent = 'This is a dynamically created image.';
21
22     // Append the new element to the container
23     container.appendChild(newElement);
24
25     // Find the button element
26     const button = document.getElementById('btn');
27
28     // Add a click event listener to the button
29     button.addEventListener('click', function () {
30       // Change the text of the dynamically created paragraph
31       newElement.textContent = 'Button clicked!';
32
33       // Create and append a new image element
34       const imgElement = document.createElement('img');
35       imgElement.setAttribute('src', 'dommani.jpg');
36       container.appendChild(imgElement);
37     });
38   </script>
39 </body>
40 </html>
```



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>CSS Manipulation</title>
7 </head>
8 <body>
9   <h1> I am Shreya</h1>
10  <h1>I'm a third year student of TSEC</h1>
11  <button onclick="myfunction()">CLICK ME TO CHANGE TEXT BACKGROUND</button>
12
13  <button onclick="myFunc()">CLICK ME TO CHANGE TEXT BACKGROUND</button>
14  <script>
15    function myFunction(){
16      document.querySelector("h1").style.backgroundColor= "yellow";
17    }
18    function myFunc(){
19      var x = document.querySelectorAll("h1");
20      x[0].style.backgroundColor = "red";
21    }
22  </script>
23 </body>
24 </html>
```







Lab Outcome : LO4- Use Javascript to develop interactive web pages.