**Batch T7**

**Practical No. 4**

**Title of Assignment: Study of Javascript and DOM**

**Student Name: Shreyash Patil**

**Student PRN: 22510039**

**Problem Statement 0: Basics of DOM**

1. What is the DOM?

Ans. The **Document Object Model (DOM)** is a programming interface for web documents. It represents the structure of a document as a tree of objects, where each object corresponds to a part of the document (like an element, attribute, or text). The DOM allows scripts to access and manipulate the document's structure, style, and content.

### 2. DOM Tree Structure

The DOM tree structure is a hierarchical representation of the HTML document. Every element, attribute, and piece of text in the document is represented as a node in this tree. The topmost node is the **Document** node, which contains **Element** nodes, **Attribute** nodes, and **Text** nodes.

#### Elements of DOM Tree Structure:

* **Document Node**: The root node, representing the entire document.
* **Element Nodes**: Represent HTML elements (like <div>, <p>, etc.).
* **Attribute Nodes**: Represent the attributes of HTML elements.
* **Text Nodes**: Represent the text content within HTML elements.
* **Comment Nodes**: Represent comments in the HTML document.

### 3. Examples

#### a. Accessing the DOM

Accessing the DOM involves retrieving an element from the DOM tree.

Example: Accessing an element by ID

let heading = document.getElementById("main-heading");

#### b. Manipulating the DOM

Manipulating the DOM involves changing the structure, content, or style of the document.

Example: Changing the text of an element

heading.textContent = "New Heading";

#### c. Event Handling

Event handling involves responding to user actions like clicks, keypresses, etc.

Example: Handling a button click

let button = document.getElementById("myButton");

button.addEventListener("click", function() {

alert("Button clicked!");

});

#### d. Traversing the DOM

Traversing the DOM involves moving from one element to another in the DOM tree.

Example: Accessing the parent node

let parent = heading.parentNode;

### 4. Performance Considerations & Browser Support

#### Performance Considerations:

* **Minimize DOM Manipulations**: Each manipulation can cause reflows and repaints, impacting performance.
* **Batch Updates**: Apply multiple updates at once to reduce reflows.
* **Use Document Fragments**: For bulk insertions, use document fragments to minimize reflows.
* **Limit DOM Depth**: A deeply nested DOM structure can slow down rendering and traversals.

#### Browser Support:

* **Cross-Browser Compatibility**: While the DOM is supported across all modern browsers, certain methods and properties may vary in behavior. Always test across different browsers.
* **Use Feature Detection**: To ensure compatibility, use feature detection techniques (e.g., if ('querySelector' in document)).

### 5. Common Methods and Properties of the DOM

#### Methods:

* getElementById(id): Returns the element with the specified ID.
* getElementsByClassName(className): Returns all elements with the specified class name.
* querySelector(selector): Returns the first element matching the CSS selector.
* createElement(tagName): Creates a new element with the specified tag.
* appendChild(node): Adds a node to the end of the list of children of a specified parent node.

#### Properties:

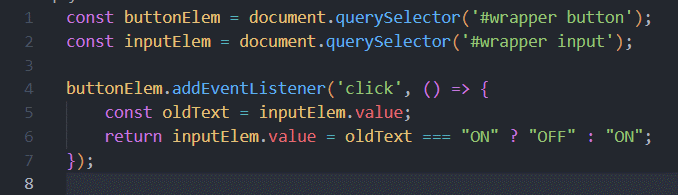
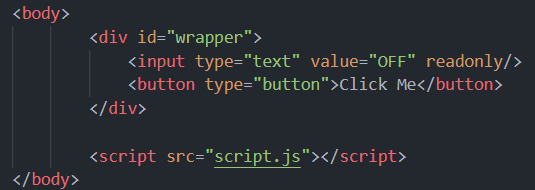
* innerHTML: Gets or sets the HTML content of an element.
* textContent: Gets or sets the text content of an element.
* style: Accesses the inline style of an element.
* classList: Provides methods to add, remove, and toggle classes.
* attributes: Returns a live collection of attributes for an element.

**Problem Statement 1: DOM selector methods:**

* **Here, the existing code expects the variables 'buttonElem' and 'inputElem' to represent the button and input elements in the example UI. Assign the respective elements to the variables. In this case, the two elements do not have unique identifiers - like for example an id. Instead they are direct descendents of a div element with id 'wrapper'. Use an appropriate selector method! Click the button to verify that the code is working.**



* **In this scenario, we are looking for a list of elements gathered in one variable - rather than only one element. Assign the list items in the view to the variable 'listItems' by using an appropriate selector method. Once you have completed the code below, verify it by hovering over the list items until all items have the value 'ON'.**

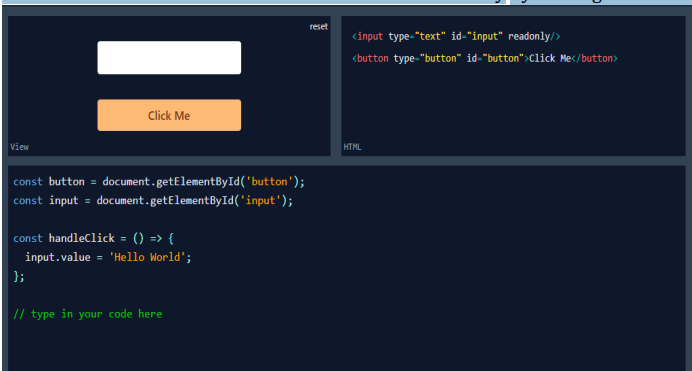


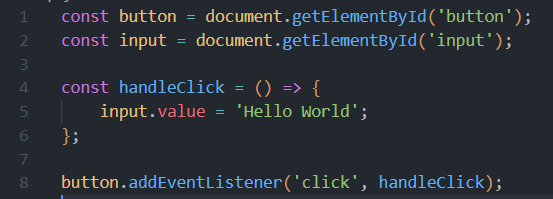
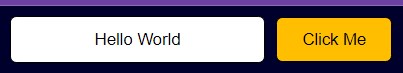
* **In this scenario, we are looking for a list of elements gathered in one variable - rather than only one element. Assign the list items in the view to the variable 'listItems' by using an appropriate selector method. Once you have completed the code below, verify it by hovering over the list items until all items have the value 'ON'.**



**Problem Statement 2: Events and user interactions**

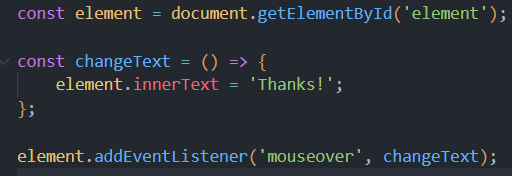
* **The Javascript function handleText fills the input field with the words Hello World. But, there is no code to execute this function. Complete the existing code below such that the function is called when the button is clicked. Verify by clicking the button.**



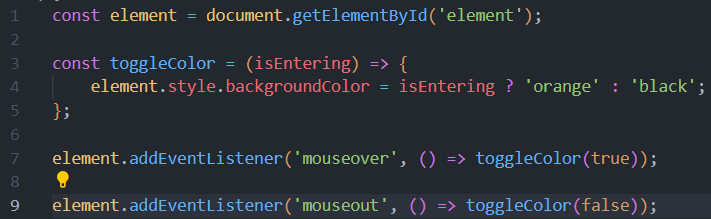
* **The Javascript function changeText changes the text inside the circle. But again, there is no code to execute this function. Complete the existing code below such that the function is called when the cursor moves onto the circle. Verify that your code works by hovering over the circle.**





* **In this scenario we want the color of the circle to change depending on the type of cursor movement. Use the function toggleColor to turn the circle orange when the cursor moves onto it. Reuse the same function to turn it black when the cursor leaves it. The tricky part is that you have to call toggleColor with different values for the parameter isEntering. Verify that your code is working by hovering the circle with the mouse cursor and leaving it again.**

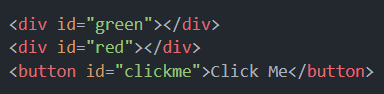
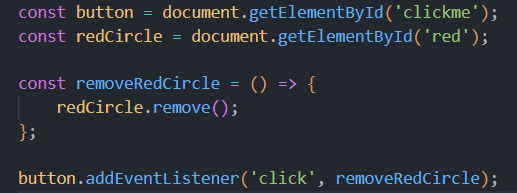




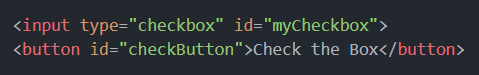
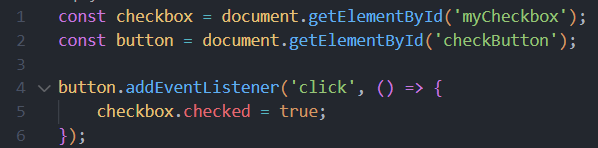
**Problem Statement 3: DOM manipulation with JavaScript**

* **Remove element from the DOM. Create t2 circles red and green and a button clickme. Place them such a way that red circle hides the green circle. Add the function removeRedCircle to remove the circle with id red from the DOM when clicked on clickme button. Make sure that you really remove the element instead of just hiding it**

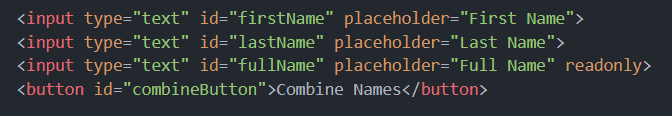
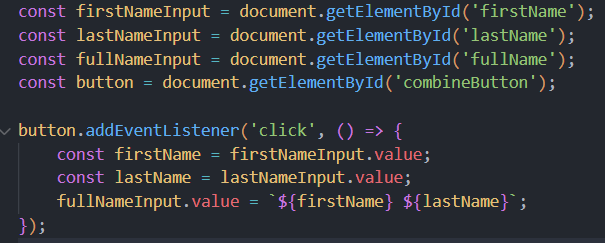
 

**Problem Statement 4: DOM fundamentals**

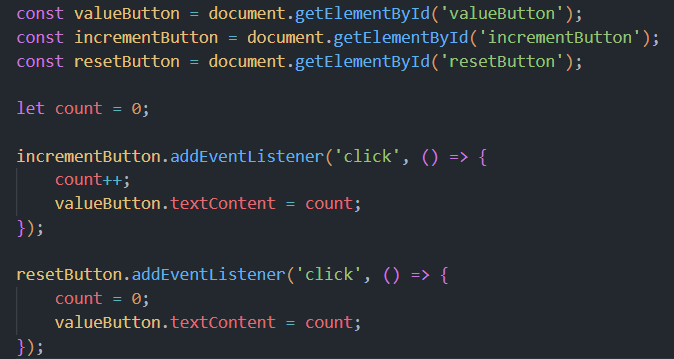
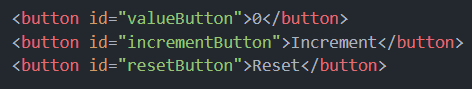
* **Create JavaScript code to interact with the displayed HTML elements. Create a checkbox and a button. Once you click the button, the checkbox should be checked.**

* **Create 3 textboxes and a button. First 2 checkboxes contain first name and last name respectively. When the button is clicked, combine the names of the first two input fields. Insert the full name in the third input field (textbox). Check if your code still works if you change the first or last name.**

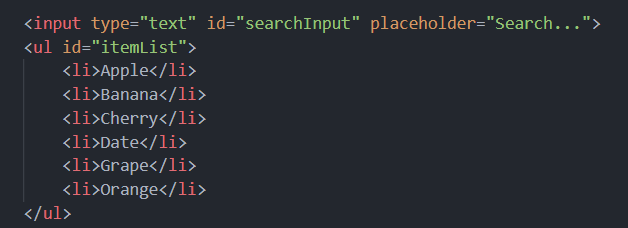
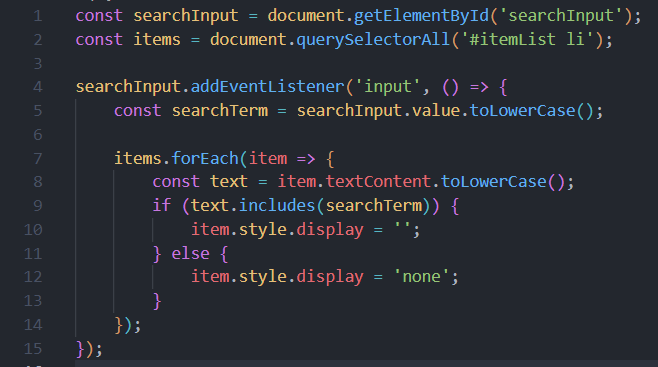
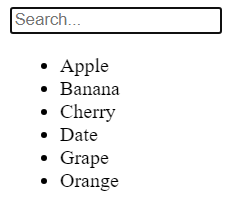
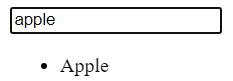
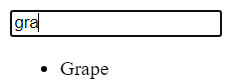
 



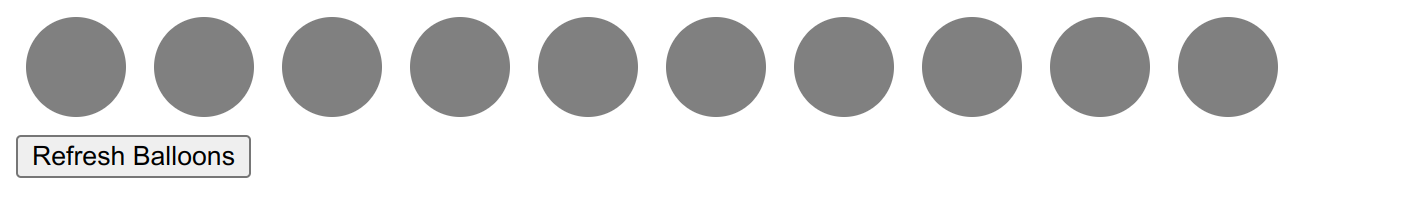
* **Create three buttons. One button displays value of 0. Other two buttons are for increment and reset. By clicking increment button each time, increase the value of the button by 1. By clicking the reset button set the value of button to 0. Confirm your code by clicking the buttons.**



**create a dynamic input filter with JavaScript. Type a search term in the input field. The displayed items in the list should match your search term. The rest of the list elements should be hidden.**

* **Create 10 balloons as shown below. Every time you hover over a balloon, it should become invisible. Your goal is to pop all the balloons one after the other. Create a refresh button. After clicking refresh button it will again display all the balloons.**

****

**Problem Statement 5: Recursive functions**

* **Create a function move that moves the button 1px to the left or the right. It is recursive because it calls itself again and again. This keeps the button moving. Extend the JavaScript code. Once you click the button, it should stop moving. When you click it again, it should move again.**

