**Batch: T5**

**Practical No. 4**

**Title of Assignment:Study of Java Script and DOM.**

**Student Name: Trishul Balaso Patil**

**Student PRN:22510051**

**Problem Statement 0: Basics of DOM**

* **What is the DOM?**
* **What is DOM Tree Structure? Elaborate its elements with example (you may**

**create DOM tree structure of previously created web pages)**

* **Give examples for following:**

**o Accessing the DOM**

**o Manipulating the DOM**

**o Event Handling**

**o Traversing the DOM**

* **What are Performance Considerations while implementing the DOM and can DOM supports all browsers?**
* **Elaborate Common Methods and Properties of DOM**

### 1. What is the DOM?

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of a document as a hierarchical tree of objects. Each object represents a part of the document, such as an element, attribute, or text. The DOM allows programs and scripts to access and manipulate the content, structure, and style of web pages dynamically.

### 2. What is the DOM Tree Structure?

The DOM Tree Structure is a way of organizing and representing the contents of a web document. It consists of nodes arranged in a hierarchical tree format:

* Document Node: The root of the tree, representing the entire document.
* Element Nodes: Each HTML tag in the document becomes an element node. For example, <html>, <body>, <div>, and <p> tags are element nodes.
* Attribute Nodes: Each attribute of an element is represented as an attribute node. For example, class="myClass" would be an attribute node of the corresponding element.
* Text Nodes: The actual text inside an element is represented as text nodes. For example, in <p>Hello</p>, "Hello" is a text node.

Example: For a simple web page with a heading and a paragraph:

html

### 3. Examples:

* Accessing the DOM:
  + Accessing an element by its ID, class, or tag name to read or modify its properties.
* Manipulating the DOM:
  + Adding, removing, or modifying elements and their attributes or content dynamically.
* Event Handling:
  + Attaching event listeners to elements to execute code in response to user actions like clicks, form submissions, or key presses.
* Traversing the DOM:
  + Navigating through the DOM tree to access parent, child, or sibling nodes.

### 4. Performance Considerations:

* Minimize Reflows and Repaints: Excessive changes to the DOM can cause performance issues due to frequent reflows and repaints.
* Batch DOM Updates: Group multiple DOM updates together to minimize the number of reflows.
* Use Efficient Selectors: Use specific and efficient selectors to reduce the time spent searching for elements.
* Avoid Inline Styles: Use CSS classes instead of modifying styles directly via JavaScript.

### 5. Browser Support:

The DOM is supported by all modern browsers, including Chrome, Firefox, Safari, Edge, and Internet Explorer. However, there may be variations in how different browsers implement or extend the DOM API, so it's important to test across multiple browsers for compatibility.

### 6. Common Methods and Properties of the DOM:

* Methods:
  + getElementById(): Returns the element with the specified ID.
  + querySelector(): Returns the first element that matches a specified CSS selector.
  + createElement(): Creates a new element node.
  + appendChild(): Adds a child node to an element.
* Properties:
  + innerHTML: Gets or sets the HTML content of an element.
  + textContent: Gets or sets the text content of an element.
  + className: Gets or sets the class attribute of an element.
  + style: Provides access to the element's inline styles.

**Problem Statement 1: DOM selector methods**

**Q.1) Here, the existing code expects the variables 'buttonElem' and 'inputElem' to**

**represent the button and input elements in the example UI. Assign the respective**

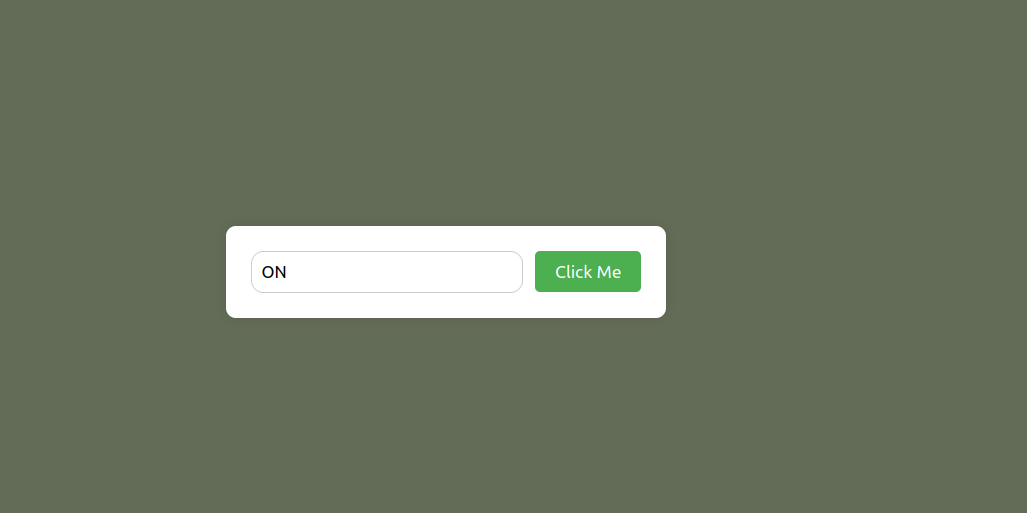
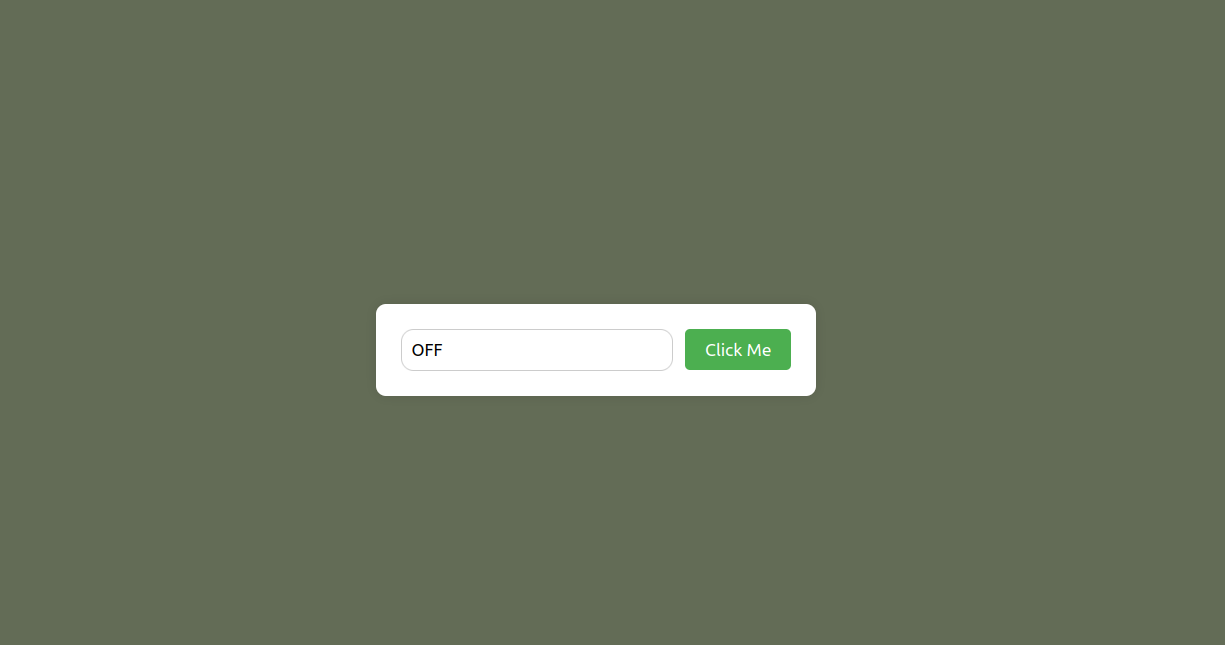
**elements to the variables. In this case, the two elements do not have unique**

**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.**

**Output:**

****

**Q.2)In this scenario, we are looking for a list of elements gathered in one variable -**

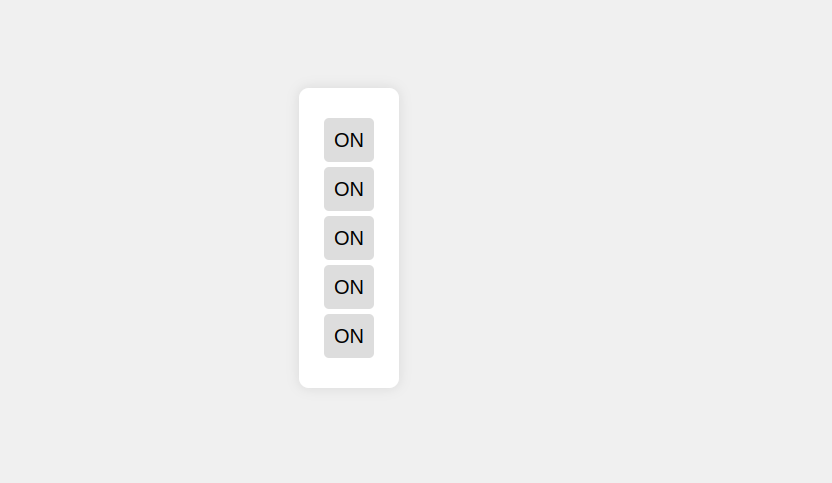
**rather than only one element. Assign the list items in the view to the variable**

**'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'**

**Output:**

****

****

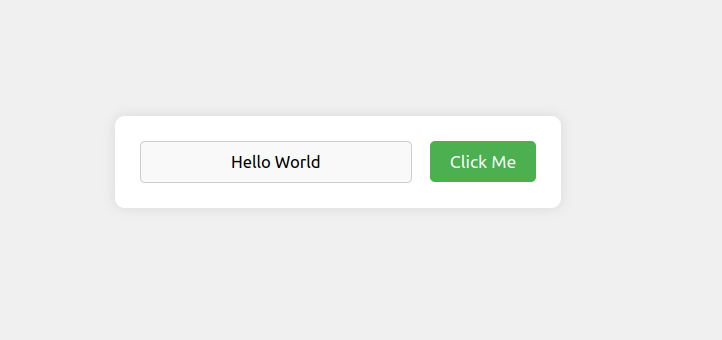
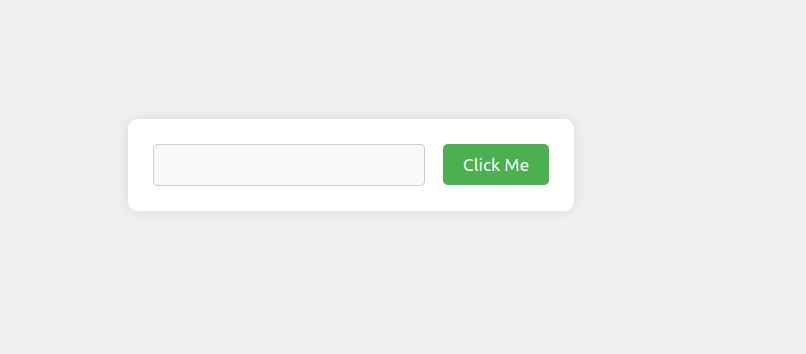
**Problem Statement 2: Events and user interactions**

**Q.1)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.**

**Output:**

****

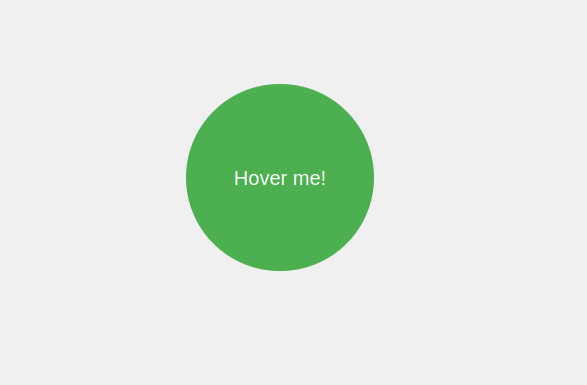
**Q.2)The Javascript function changeText changes the text inside the circle. But again,**

**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.**

**Output:**

****

**Q.3) In this scenario we want the color of the circle to change depending on the type of**

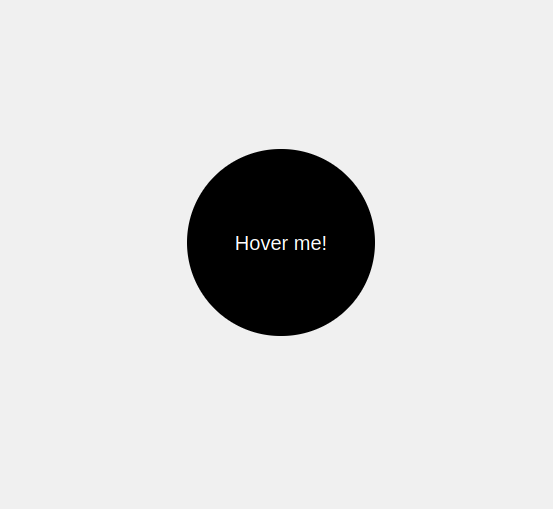
**cursor movement. Use the function toggleColor to turn the circle orange when the**

**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.**

**Output:**

**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.**

**Output:**

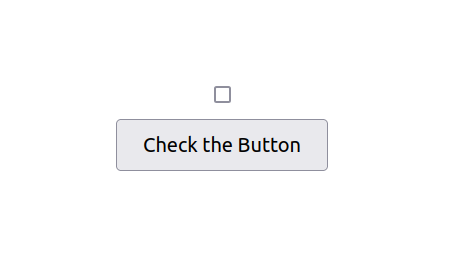
****

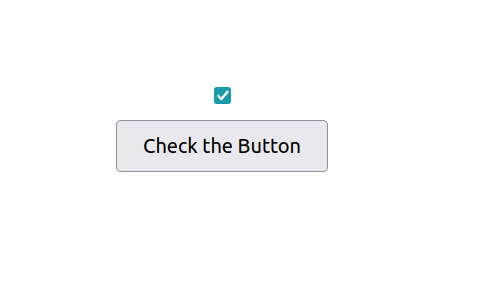
****

**Problem Statement 4: DOM fundamentals**

**Q.1)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.**

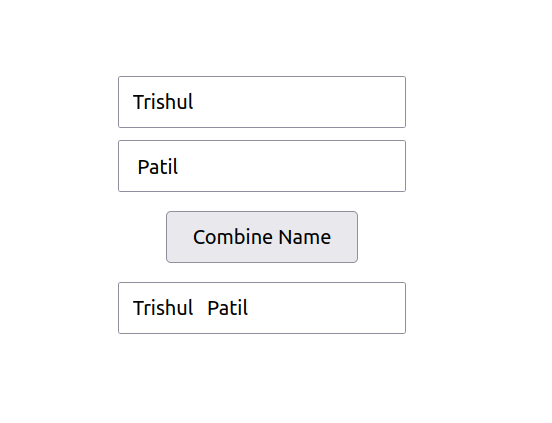
**Output:**

****

****

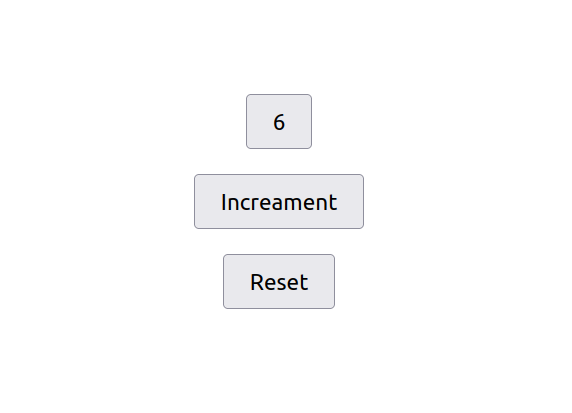
**Q.2) Create 3 textboxes and a button. First 2 checkboxes contain first name and last name respectively. When the button is clicked, combine the names of the first two input fields. Insert the full name in the third input field (textbox). Check if your code still works if you change the first or last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

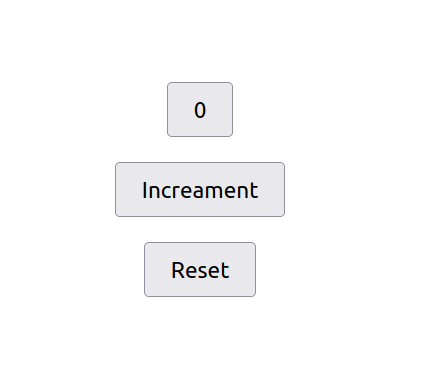
**Output:**

****

**Q.3) 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.**

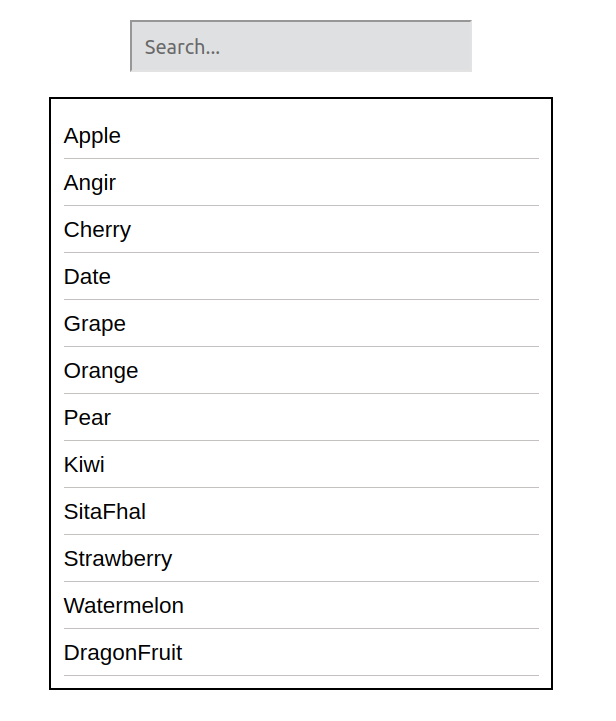
**Output:**

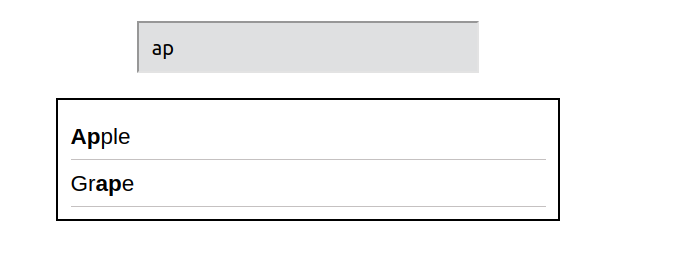
****

****

**Q.4) 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.**

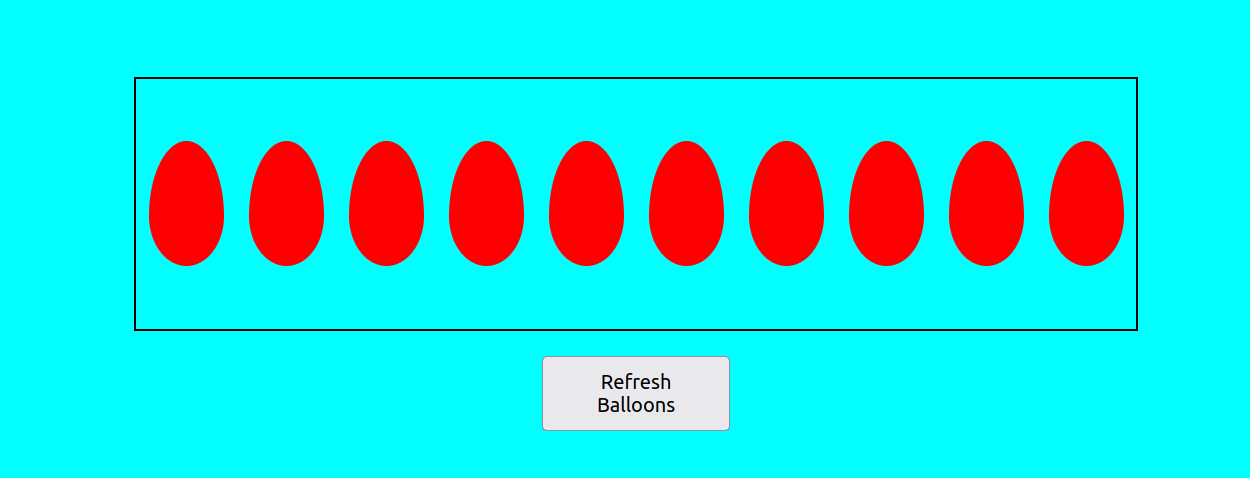
**Output:**

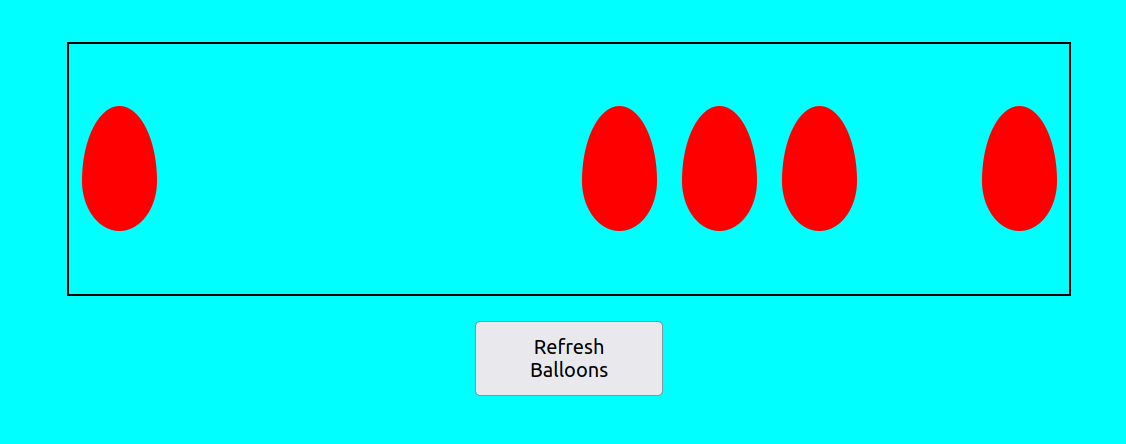
****

****

**Q.5) 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.**

**Output:**

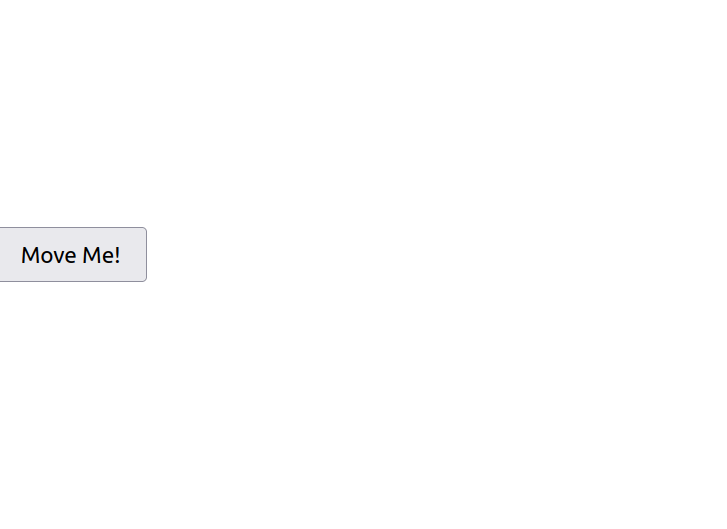
****

****

**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.**

**Output:**

****