Batch: T6

Practical No. 4

Title of Assignment: Study of Java Script and DOM

Student Name: Parshwa Herwade

Student PRN: 22510064

Problem Statement 0: Basics of DOM

What is the DOM?

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of a web page as a tree-like data structure, allowing developers to interact with and manipulate the page's content, structure, and style.

What is DOM Tree Structure?

The DOM Tree Structure is a hierarchical representation of an HTML document, where each node represents an element, attribute, or text content. The tree structure consists of:

- Elements: Represented by tags (e.g., <div>, <p>, etc.)
- Attributes: Represented by key-value pairs (e.g., id="header", class="container", etc.)
- Text Content: Represented by text nodes (e.g., "Hello World!")

Example DOM Tree Structure:

```
<html>
 <head>
  <title>Page Title</title>
 </head>
 <body>
```

```
    <div id="header">
      <h1>Heading</h1>
    </div>
    <p>Paragraph text</p>
  </body>
</html>
```

Elements of the DOM Tree Structure:

- Root Node (<html>): The topmost node of the tree
- Element Nodes (<head>, <body>, <div>, etc.): Represent HTML elements
- Attribute Nodes (id="header", class="container", etc.): Represent element attributes
- Text Nodes ("Page Title", "Heading", "Paragraph text", etc.): Represent text content

Accessing the DOM:

- document.getElementById('id'): Retrieves an element by its ID
- document.querySelector('selector'): Retrieves an element by its CSS selector
- document.getElementsByTagName('tag'): Retrieves a collection of elements by their tag name

Manipulating the DOM:

- element.textContent = 'new text': Changes the text content of an element
- element.setAttribute('attribute', 'value'): Sets an attribute on an element
- element.appendChild(newElement): Adds a new element to an existing element

Event Handling:

- element.addEventListener('event', function() { ... }): Attaches an event listener to an element

- element.removeEventListener('event', function() { ... }): Removes an event listener from an element

Traversing the DOM:

- element.parentNode: Retrieves the parent node of an element
- element.childNodes: Retrieves a collection of child nodes of an element
- element.nextSibling: Retrieves the next sibling node of an element

Performance Considerations:

- DOM Manipulation: Minimize direct DOM manipulation to improve performance
- Caching: Cache frequently accessed DOM elements to reduce lookup times
- Event Delegation: Use event delegation to reduce the number of event listeners

Common Methods and Properties of DOM:

- document.createElement('tag'): Creates a new element
- element.cloneNode(): Clones an element
- element.removeChild(child): Removes a child element
- element.offsetWidth: Retrieves the width of an element
- element.offsetHeight: Retrieves the height of an element

Browser Support:

The DOM is supported by all modern web browsers, including:

- Google Chrome

- Mozilla Firefox

- Microsoft Edge

- Safari

- Opera

Problem Statement 1: DOM selector methods

- Technologies/Frameworks: HTML, CSS, JavaScript, DOM, CSS Selectors
- Related Theory:
  - CSS Selectors: Used to select elements in an HTML document based on their attributes, classes, IDs, etc.
  - DOM Querying: Methods used to retrieve elements from the DOM, such as querySelector and querySelectorAll.

Problem Statement 2: Events and user interactions

- Technologies/Frameworks: HTML, CSS, JavaScript, DOM, Event Listeners
- Related Theory:
  - Event Listeners: Functions that are called when a specific event occurs on an element.
  - Event Propagation: The process by which events are passed from a child element to its parent elements.

Problem Statement 3: DOM manipulation with JavaScript

- Technologies/Frameworks: HTML, CSS, JavaScript, DOM
- Related Theory:
  - DOM Manipulation: Changing the structure or content of an HTML document using JavaScript.
  - Element Creation: Creating new elements using document.createElement.
  - Element Removal: Removing elements from the DOM using element.remove.

Problem Statement 4: DOM fundamentals

- Technologies/Frameworks: HTML, CSS, JavaScript, DOM

- Related Theory:

   - DOM Tree: A hierarchical representation of an HTML document, where each node represents an element or attribute.

   - DOM Events: Events that occur when a user interacts with a web page, such as clicks, hover, etc.

   - DOM Manipulation: Changing the structure or content of an HTML document using JavaScript.

Problem Statement 5: Recursive functions

- Technologies/Frameworks: JavaScript

- Related Theory:

   - Recursion: A programming technique where a function calls itself repeatedly until a base case is reached.

   - Function Closures: Functions that have access to their own scope and can be used to create recursive functions.

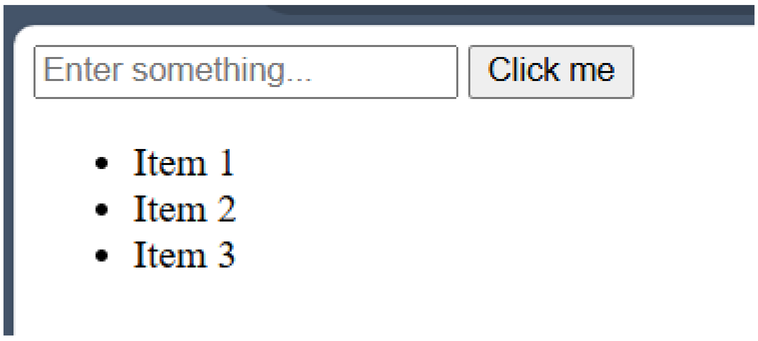
<u>Perform following problem statements for DOM using Javascript</u>

<u>Problem Statement 1: DOM selector methods</u>

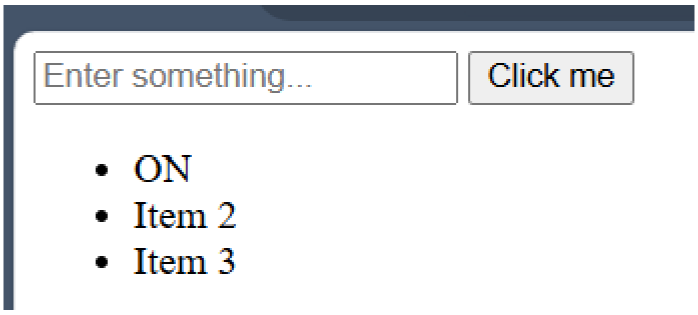
 Here, the existing code expects the variables 'buttonElem' and 'inputElem' to represent the button and input elements in the example UI. Assign the respective elements to the variables. In this case, the two elements do not have unique identifiers - like for example an id. Instead they are direct descendents 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'

ANS.



- Item 1
- Item 2
- Item 3



- ON
- Item 2
- Item 3

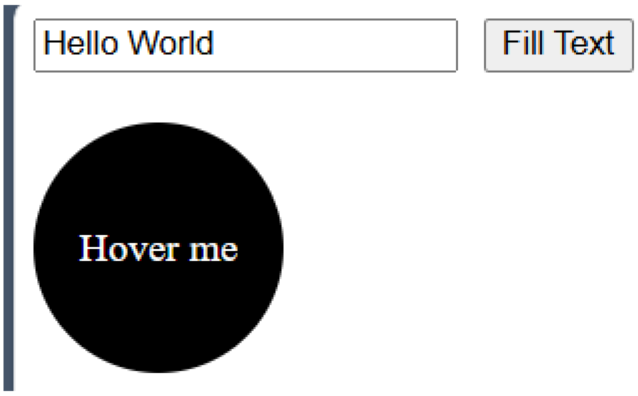
Problem Statement 2: Events and user interactions

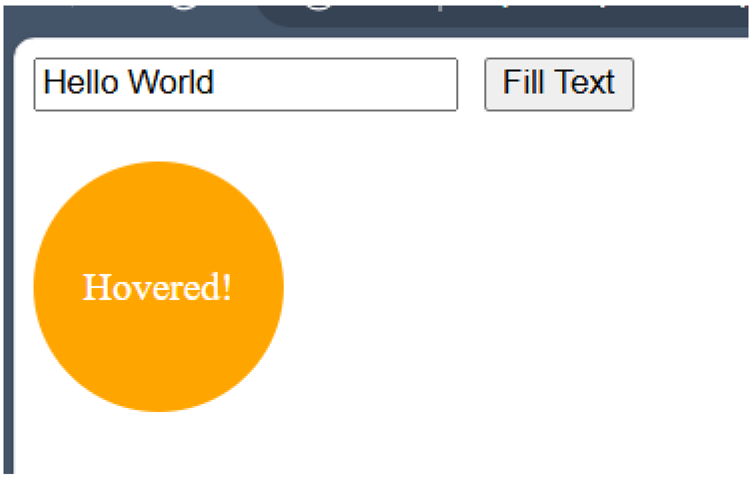
 The Javascript function handleText fills the input field with the words Hello World. But, there is no code to execute this function. Complete the existing code below such that the function is called when the button is clicked. Verify by clicking the button.

 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.

ANS.

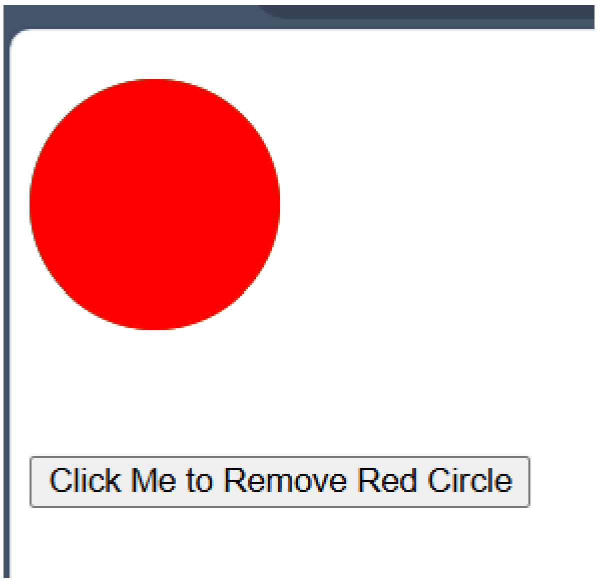


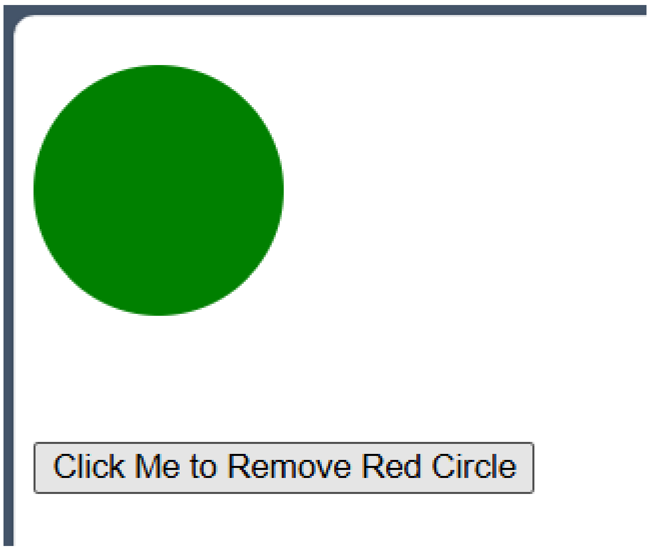


Problem Statement 3: DOM manipulation with JavaScript

 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.

ANS.





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.

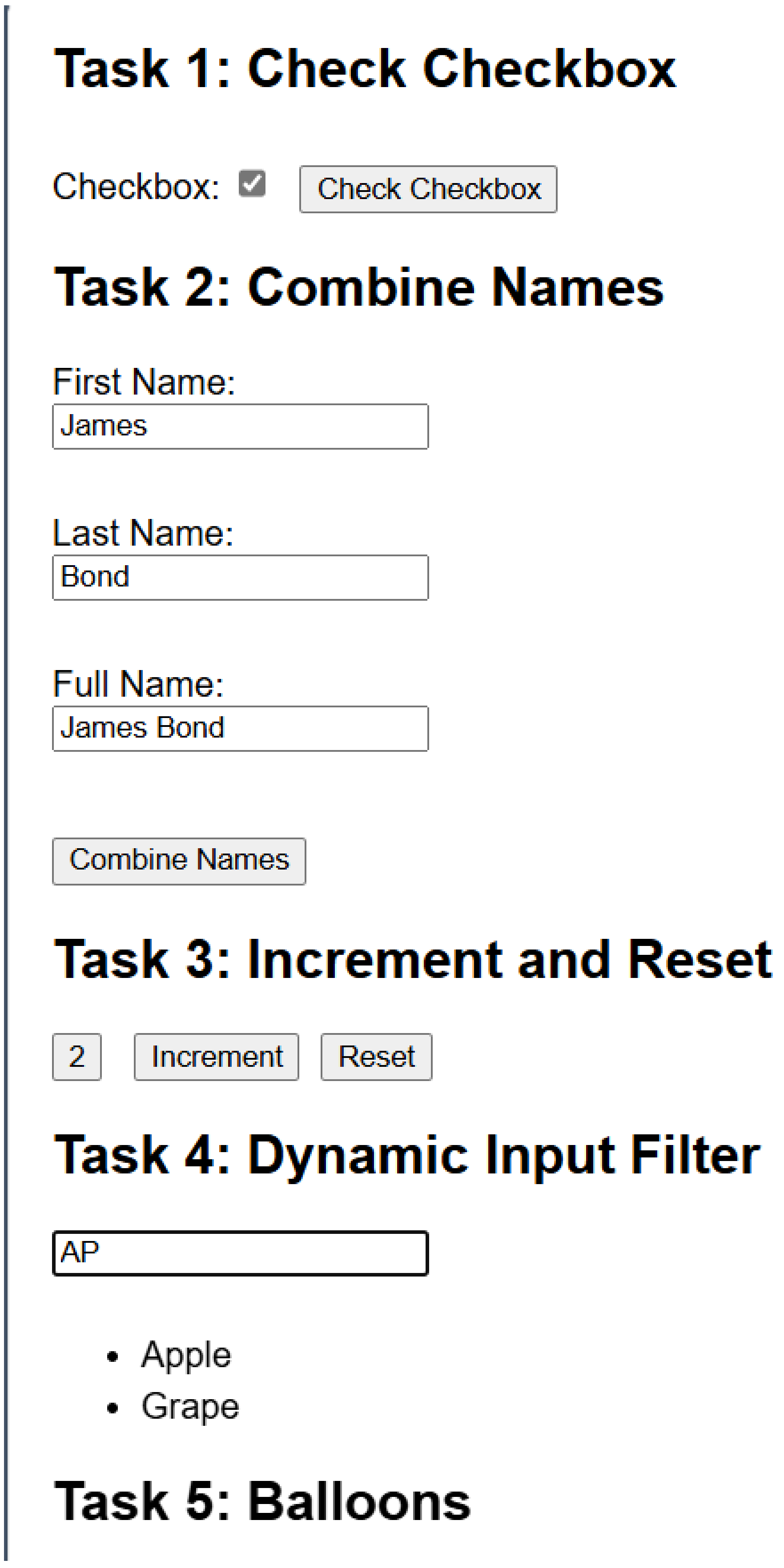
 Create 3 textboxes and a button. First 2 checkboxes contain first name and last name respectively. When the button is clicked, combine the names of the first two input fields. Insert the full name in the third input field (textbox). Check if your code still works if you change the first or last name.

 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.

ANS.



Task 1: Check Checkbox

Checkbox: ✓

Task 2: Combine Names

First Name: James

Last Name: Bond

Full Name: James Bond

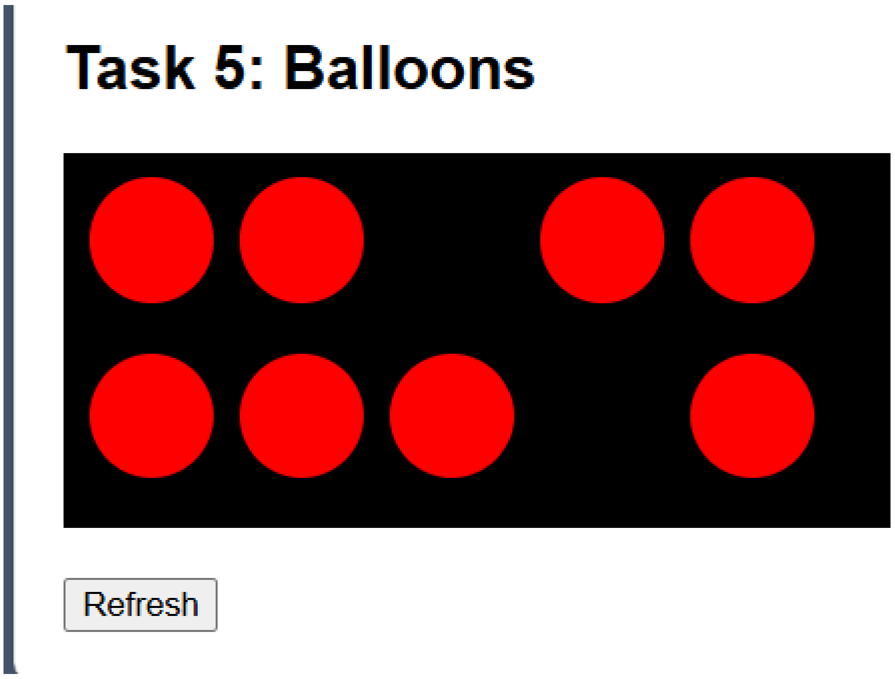
Task 3: Increment and Reset

Task 4: Dynamic Input Filter

AP

- Apple
- Grape

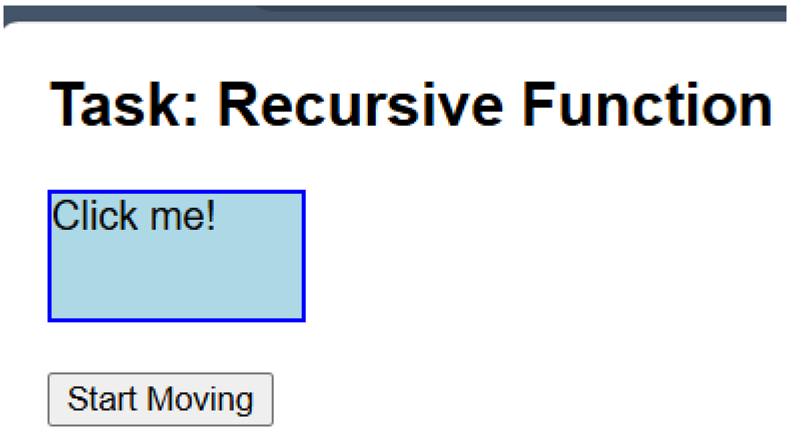
Task 5: Balloons

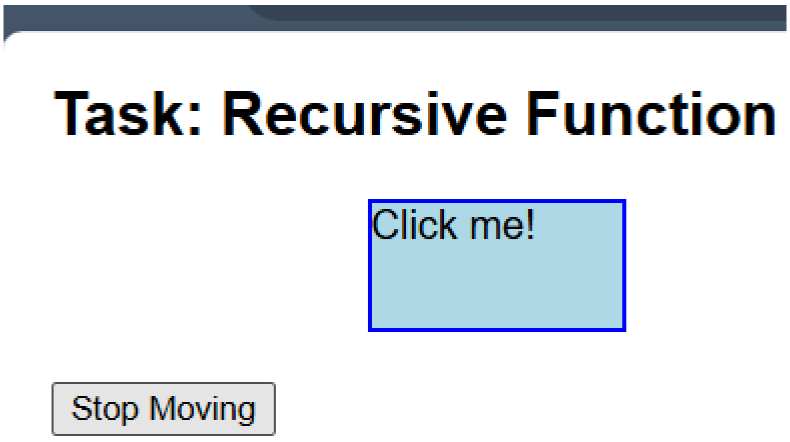


Problem Statement 5: Recursive functions

 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.

ANS.





Task: Recursive Function