**Batch T4**

**Practical No. 4**

**Title of Assignment: DOM**

**Student Name: Daivik Liladhar Karbhari**

**Student PRN: 22510111**

**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 page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects; that way, programming languages can interact with the page.

**What is DOM Tree Structure?**

The DOM tree structure is a hierarchical representation of the elements in an HTML or XML document. Each element, attribute, and piece of text in the document is represented by a node in the tree. The tree starts with a single root node, which branches out to child nodes, and those child nodes can have their own children, forming a tree-like structure.

**Example:**

**Consider the following HTML:**

<!DOCTYPE html>

<html>

<head>

<title>Sample Page</title>

</head>

<body>

<h1>Hello, World!</h1>

<p>This is a sample page.</p>

</body>

</html>

The DOM tree structure for this HTML would look like this:

Document

├── html

├── head

│ └── title

│ └── "Sample Page"

└── body

├── h1

│ └── "Hello, World!"

└── p

└── "This is a sample page."

**Examples**

**Accessing the DOM**

// Accessing an element by ID

const element = document.getElementById('myElement');

// Accessing elements by class name

const elements = document.getElementsByClassName('myClass');

// Accessing elements by tag name

const paragraphs = document.getElementsByTagName('p');

**Manipulating the DOM**

// Changing the content of an element

const element = document.getElementById('myElement');

element.textContent = 'New Content';

// Adding a new element

const newElement = document.createElement('div');

newElement.textContent = 'I am a new div';

document.body.appendChild(newElement);

// Removing an element

const elementToRemove = document.getElementById('removeMe');

elementToRemove.parentNode.removeChild(elementToRemove);

**Event Handling**

// Adding an event listener

const button = document.getElementById('myButton');

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

alert('Button was clicked!');

});

**Traversing the DOM**

// Accessing parent node

const child = document.getElementById('childElement');

const parent = child.parentNode;

// Accessing child nodes

const parent = document.getElementById('parentElement');

const children = parent.childNodes;

// Accessing sibling nodes

const element = document.getElementById('myElement');

const nextSibling = element.nextSibling;

const previousSibling = element.previousSibling;

**Performance Considerations**

Minimize DOM Access: Accessing and manipulating the DOM can be slow. Minimize the number of times you access the DOM.

Batch DOM Updates: Make changes to the DOM in batches rather than one at a time.

Use Document Fragments: When adding multiple elements, use a document fragment to minimize reflows and repaints.

Debounce Events: For events that fire frequently (like scroll or resize), debounce the event handler to improve performance.

**Browser Support**

The DOM is supported by all modern browsers, but there can be differences in implementation and support for specific methods and properties. Always check for compatibility and consider using polyfills for unsupported features.

**Common Methods and Properties of DOM**

**Methods**

getElementById(id): Returns the element with the specified ID.

getElementsByClassName(className): Returns a collection of elements with the specified class name.

getElementsByTagName(tagName): Returns a collection of elements with the specified tag name.

querySelector(selector): Returns the first element that matches the specified CSS selector.

querySelectorAll(selector): Returns a collection of elements that match the specified CSS selector.

createElement(tagName): Creates a new element with the specified tag name.

appendChild(node): Adds a new child node to an element.

removeChild(node): Removes a child node from 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: Gets or sets the inline style of an element.

parentNode: Returns the parent node of an element.

childNodes: Returns a collection of an element's child nodes.

firstChild: Returns the first child node of an element.

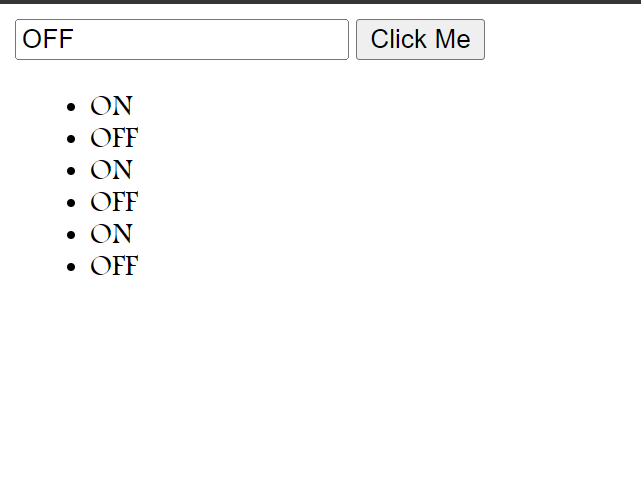
lastChild: Returns the last child node of an element.

nextSibling: Returns the next sibling node of an element.

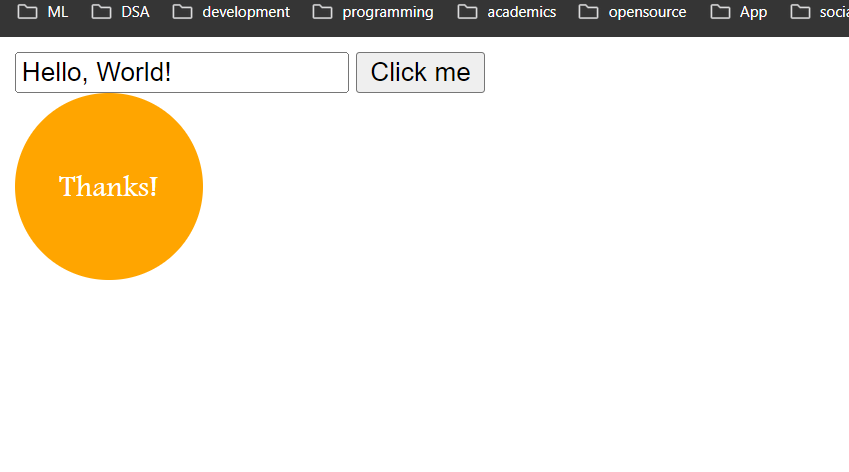
previousSibling: Returns the previous sibling node of an element.

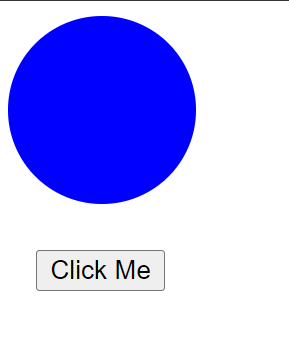
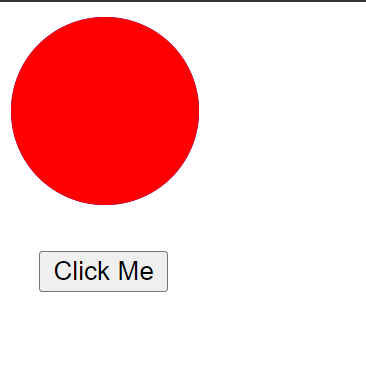
**Problem Statement 1: DOM selector methods**

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

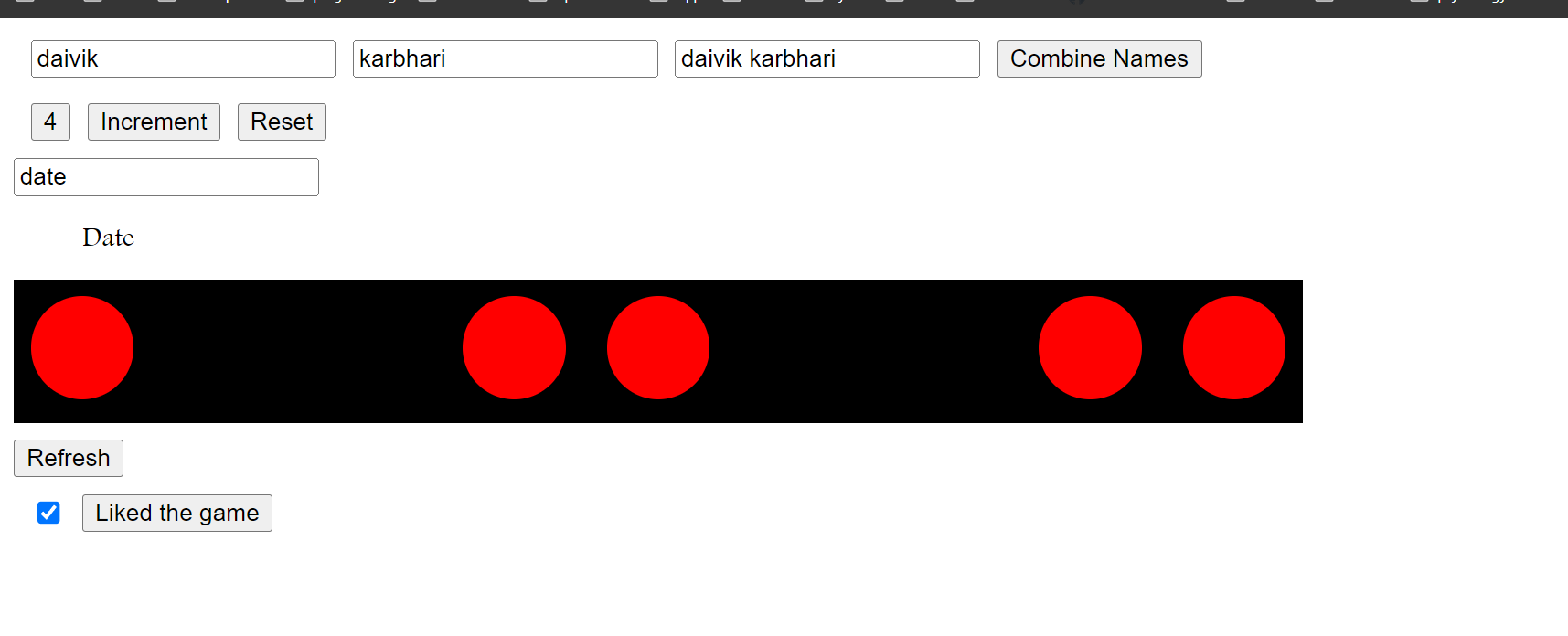


**Problem Statement 3: DOM manipulation with JavaScript**





**Problem Statement 4**: **DOM fundamentals**



**Problem Statement 5: Recursive functions**

