

## Lesson 02 Demo 06

### Implementing CRUD Operations on a Doubly Linked List

**Objective:** To create a doubly linked list in JavaScript with CRUD functionalities such as node addition, traversal, value modification, and deletion to enhance your understanding of bidirectional data structures

**Tools required:** Visual Studio Code (VS Code) and JavaScript

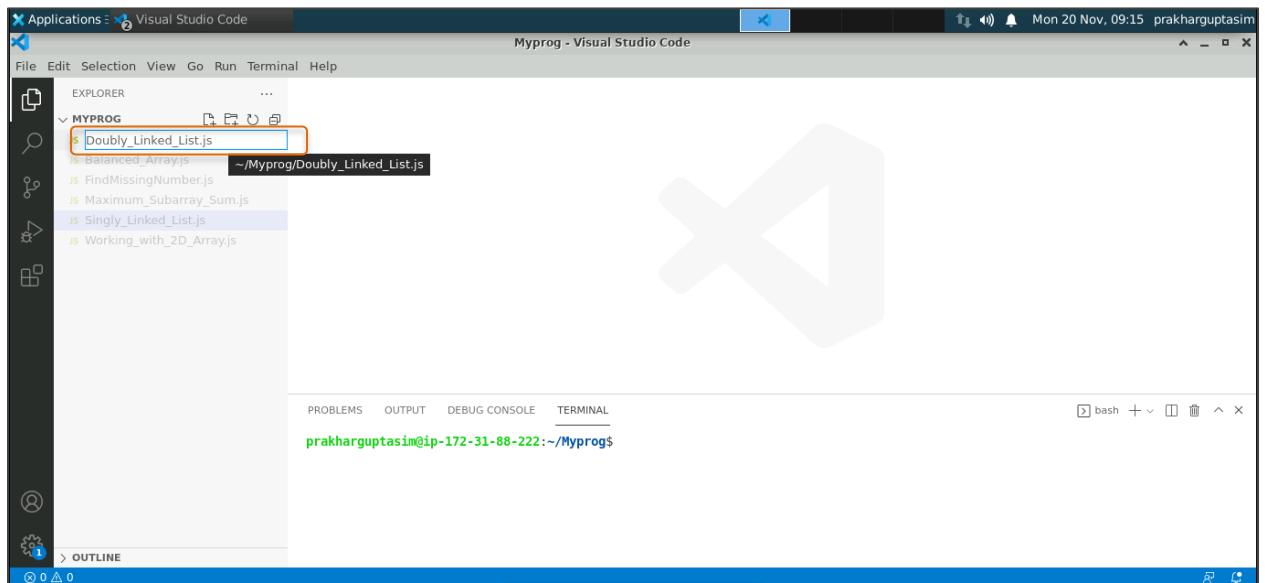
**Prerequisites:** Completion of Lesson 02 Demo 01

Steps to be followed:

1. Create a JavaScript file and execute it

#### Step 1: Create a JavaScript file and execute it

1.1 Open the Visual Studio Code editor and create a JavaScript file named **Doubly\_Linked\_List.js**



1.2 Add the following code to the file:

```
class ListNode {  
    constructor(data) {  
        this.data = data;  
        this.next = null;  
        this.prev = null;  
    }  
}  
  
class DoublyLinkedList {  
    constructor() {  
        this.head = null;  
        this.tail = null;  
    }  
  
    // Create: Add a new node to the end of the list  
    add(data) {  
        const newNode = new ListNode(data);  
        if (!this.head) {  
            this.head = newNode;  
            this.tail = newNode;  
        } else {  
            newNode.prev = this.tail;  
            this.tail.next = newNode;  
            this.tail = newNode;  
        }  
    }  
  
    // Read: Traverse and display elements of the list  
    read() {  
        let current = this.head;  
        while (current) {  
            console.log(current.data);  
            current = current.next;  
        }  
    }  
}
```

```
// Update: Modify the value of a node at a given position
update(position, data) {
    let current = this.head;
    let count = 0;
    while (current) {
        if (count === position) {
            current.data = data;
            return;
        }
        current = current.next;
        count++;
    }
    console.log("Position not found");
}

// Delete: Remove a node from the list at a specified position
delete(position) {
    if (position === 0) {
        this.head = this.head.next;
        if (this.head) {
            this.head.prev = null;
        } else {
            this.tail = null;
        }
        return;
    }

    let current = this.head;
    let count = 0;

    while (current) {
        if (count === position) {
            if (current.next) {
                current.next.prev = current.prev;
            } else {
                this.tail = current.prev;
            }
            if (current.prev) {
```

```

        current.prev.next = current.next;
    } else {
        this.head = current.next;
    }
    return;
}
current = current.next;
count++;
}
console.log("Position not found");
}
}

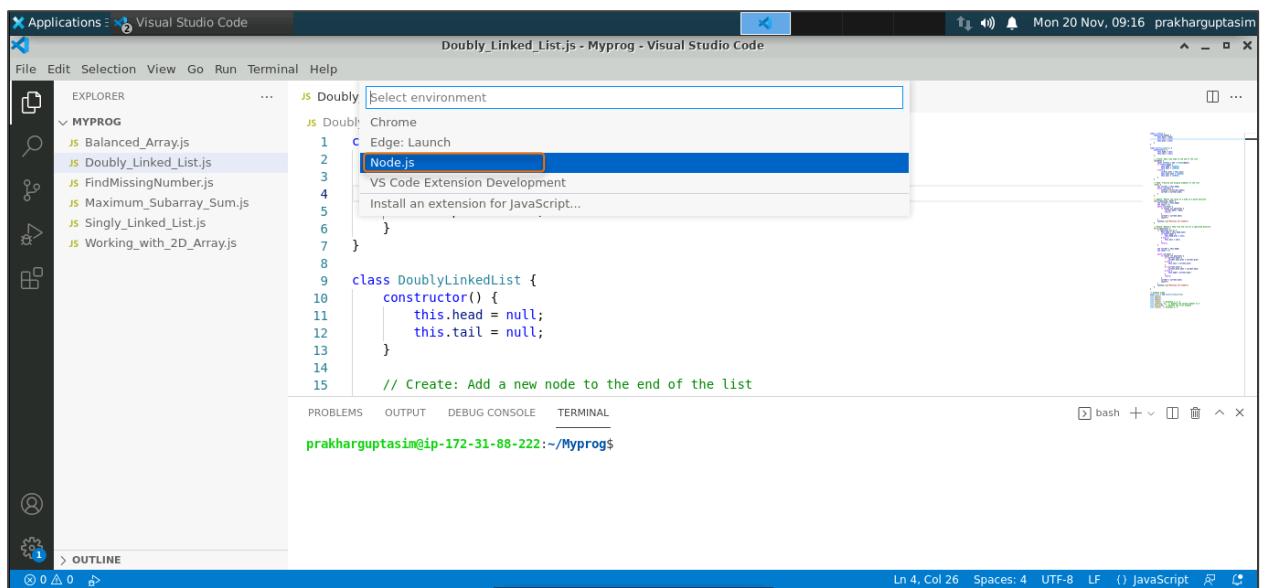
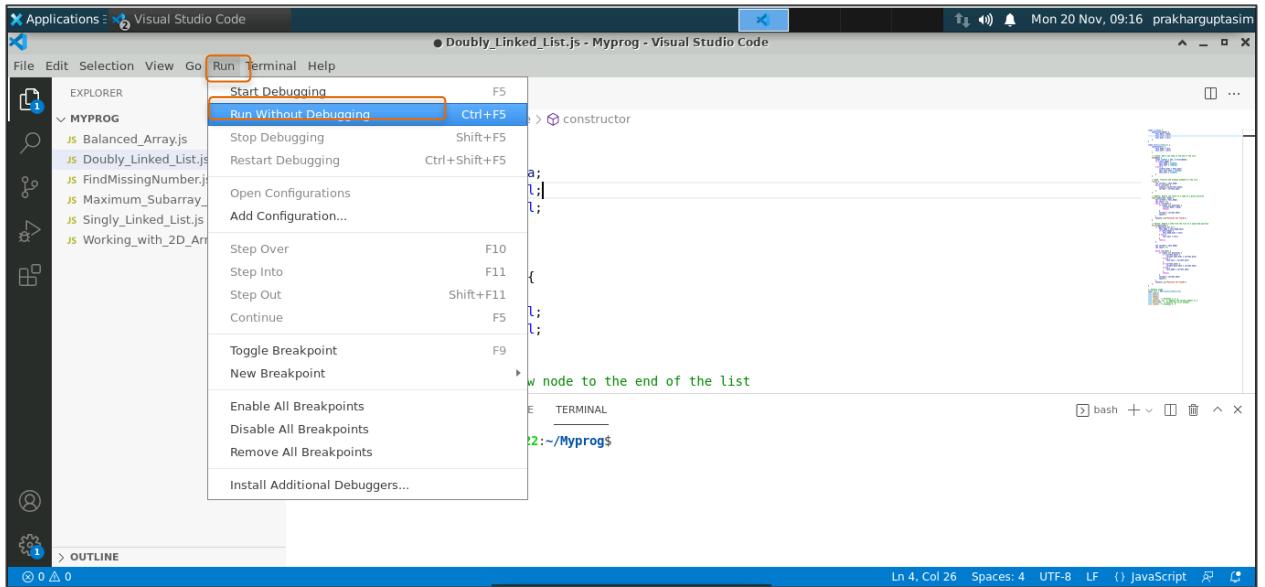
// Example usage
const list = new DoublyLinkedList();
list.add(1);
list.add(2);
list.add(3);
list.read(); // Displays 1, 2, 3
list.update(1, 4); // Updates the second element to 4
list.delete(0); // Deletes the first element
list.read(); // Displays 4, 3

```

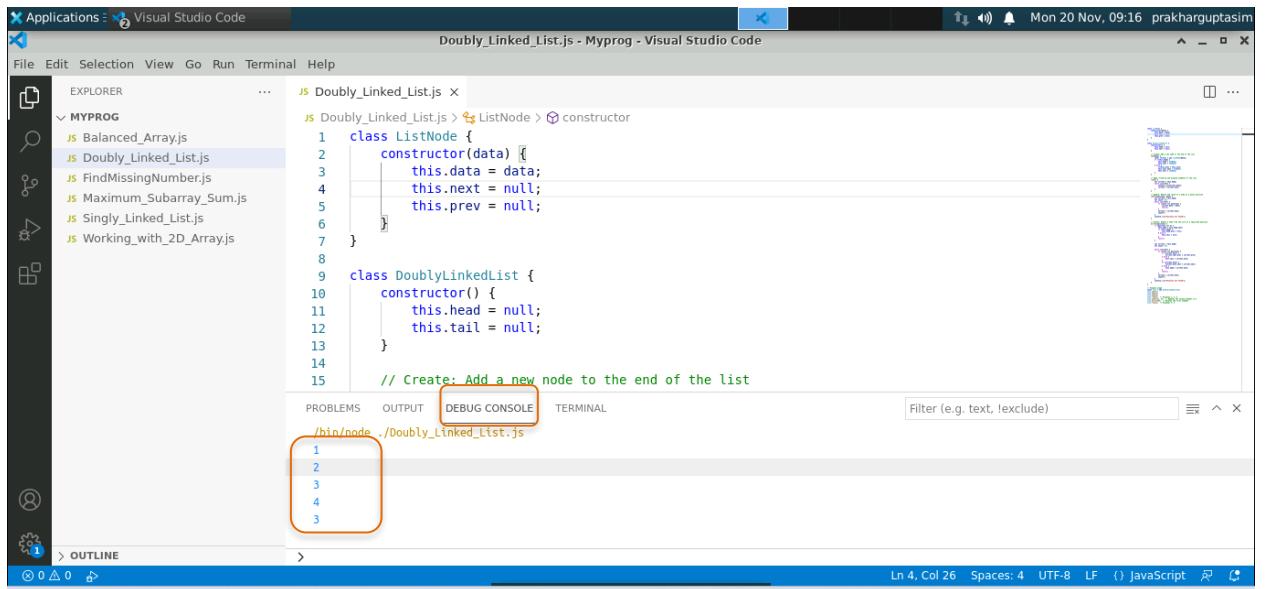
The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Applications - Visual Studio Code, Doubly\_Linked\_List.js - Myprog - Visual Studio Code, Mon 20 Nov, 09:15 prakharguptasim
- File Explorer:** Shows a folder named "MYPROG" containing several JavaScript files: Balanced\_Arrays.js, Doubly\_Linked\_List.js (selected), FindMissingNumber.js, Maximum\_Subarray\_Sum.js, Singly\_Linked\_List.js, and Working\_with\_2D\_Array.js.
- Code Editor:** Displays the content of Doubly\_Linked\_List.js. The code defines a DoublyLinkedList class with a constructor that initializes head and tail pointers to null. It also defines a ListNode class with a constructor that initializes data, next, and prev pointers to null. A comment at the bottom suggests adding a new node to the end of the list.
- Terminal:** Shows the command prompt: prakharguptasim@ip-172-31-88-222:~/Myprog\$
- Status Bar:** Shows line 4, column 26, spaces: 4, UTF-8, LF, JavaScript, and other icons.

1.3 Click **Run** and then **Run Without Debugging**. Select **Node.js** to check the output in the DEBUG CONSOLE.



#### 1.4 View the output in the **DEBUG CONSOLE** as shown below:



The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Applications > Visual Studio Code - Doubly\_Linked\_List.js - Myprog - Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer:** Shows files in the 'MYPROG' folder: Balanced\_Array.js, Doubly\_Linked\_List.js, FindMissingNumber.js, Maximum\_Subarray\_Sum.js, Singly\_Linked\_List.js, Working\_with\_2D\_Array.js.
- Code Editor:** Displays the contents of Doubly\_Linked\_List.js, defining ListNode and DoublyLinkedList classes.
- Bottom Navigation:** PROBLEMS, OUTPUT, DEBUG CONSOLE (highlighted), TERMINAL.
- Output Area:** Shows the command '/bin/node ./Doubly\_Linked\_List.js' followed by the output:

```
1
2
3
4
5
```

The output '5' is highlighted with an orange box.
- Status Bar:** Ln 4, Col 26 Spaces: 4 UTF-8 LF () JavaScript

By following these steps, you have successfully performed CRUD operations on a doubly linked list using JavaScript. This includes adding new nodes, traversing the list, updating node values, and deleting nodes, which are key operations for managing and manipulating bidirectional data structures.