

Lesson 02 Demo 06

Implementing CRUD Operations on a Circular Linked List

Objective: To implement a circular linked list in JavaScript, with CRUD functionalities including node addition, traversal, value modification, and node deletion to strengthen your understanding of circular data structure operations

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

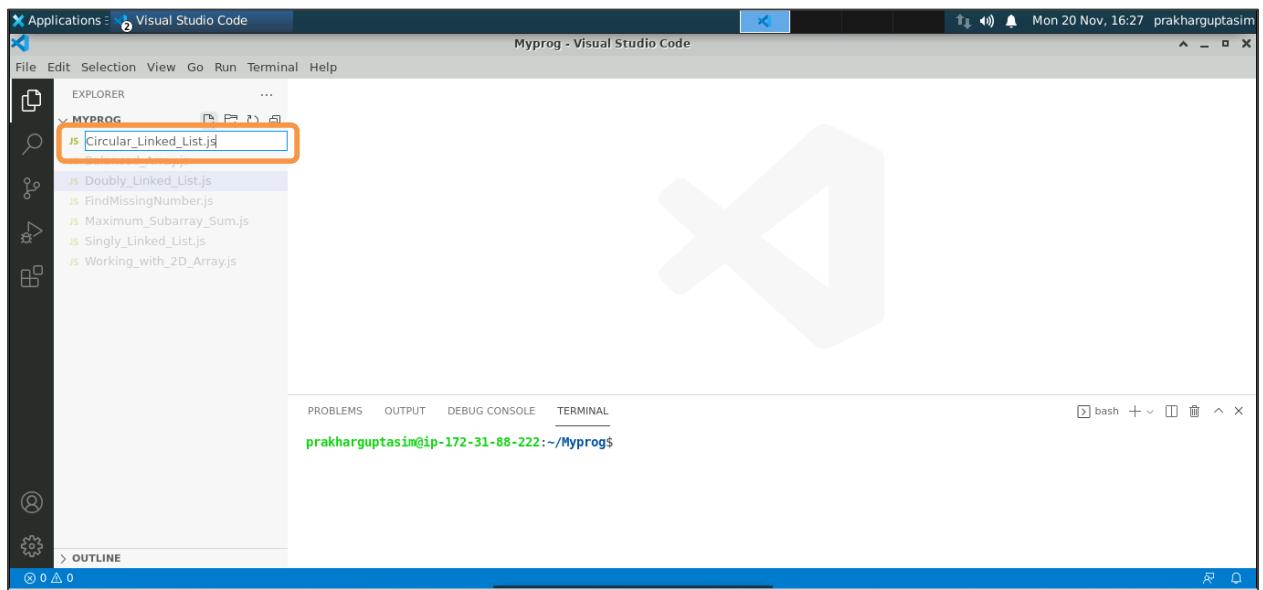
Prerequisites: Completion of Lesson 02 Demo 01

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 **Circular_Linked_List.js**



1.2 Add the following code to the file:

```
class ListNode {
    constructor(data) {
        this.data = data;
        this.next = null;
    }
}

class CircularLinkedList {
    constructor() {
        this.head = null;
    }

    // Create: Add a new node to the list
    add(data) {
        const newNode = new ListNode(data);
        if (!this.head) {
            this.head = newNode;
            newNode.next = this.head;
        } else {
            let current = this.head;
            while (current.next !== this.head) {
                current = current.next;
            }
            current.next = newNode;
            newNode.next = this.head;
        }
    }
}

// Read: Traverse and display elements of the list
read() {
    if (!this.head) {
        return;
    }
    let current = this.head;
    do {
        console.log(current.data);
```

```

        current = current.next;
    } while (current !== this.head);
}

// Update: Modify the value of a node at a given position
update(position, data) {
    if (!this.head) {
        return;
    }

    let current = this.head;
    let count = 0;
    do {
        if (count === position) {
            current.data = data;
            return;
        }
        current = current.next;
        count++;
    } while (current !== this.head);

    console.log("Position not found");
}

// Delete: Remove a node from the list at a specified position
delete(position) {
    if (!this.head) {
        return;
    }

    if (position === 0) {
        if (this.head.next === this.head) {
            this.head = null;
        } else {
            let current = this.head;
            while (current.next !== this.head) {
                current = current.next;
            }
        }
    }
}

```

```
        this.head = this.head.next;
        current.next = this.head;
    }
    return;
}

let current = this.head;
let previous = null;
let count = 0;
do {
    if (count === position) {
        previous.next = current.next;
        return;
    }
    previous = current;
    current = current.next;
    count++;
} while (current !== this.head && current !== null);

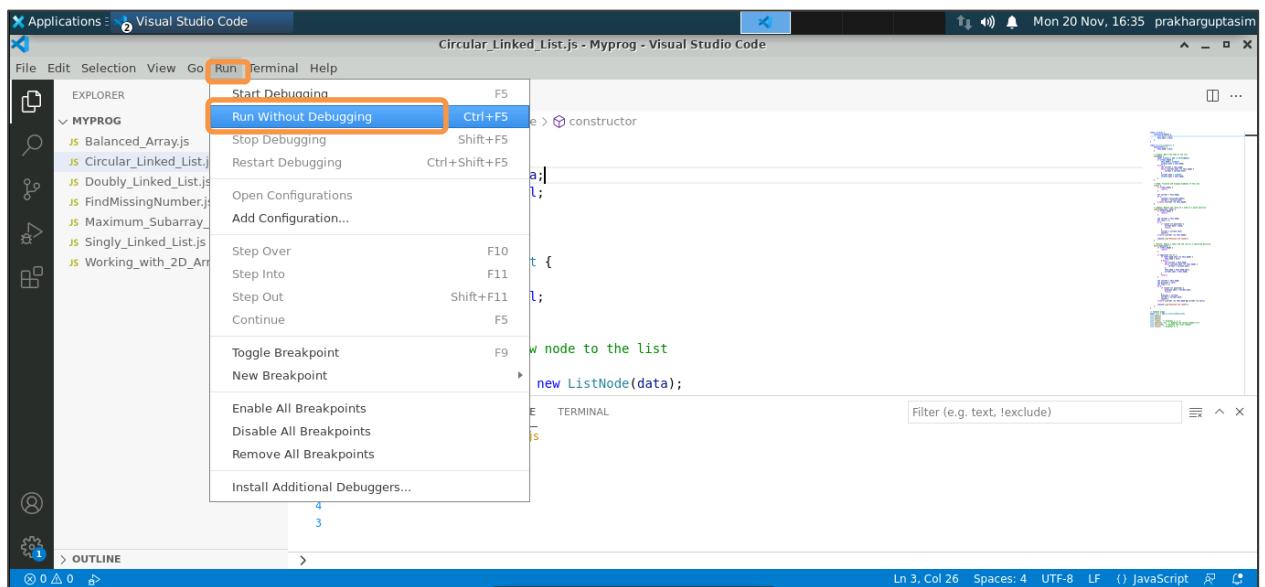
console.log("Position not found");
}
}

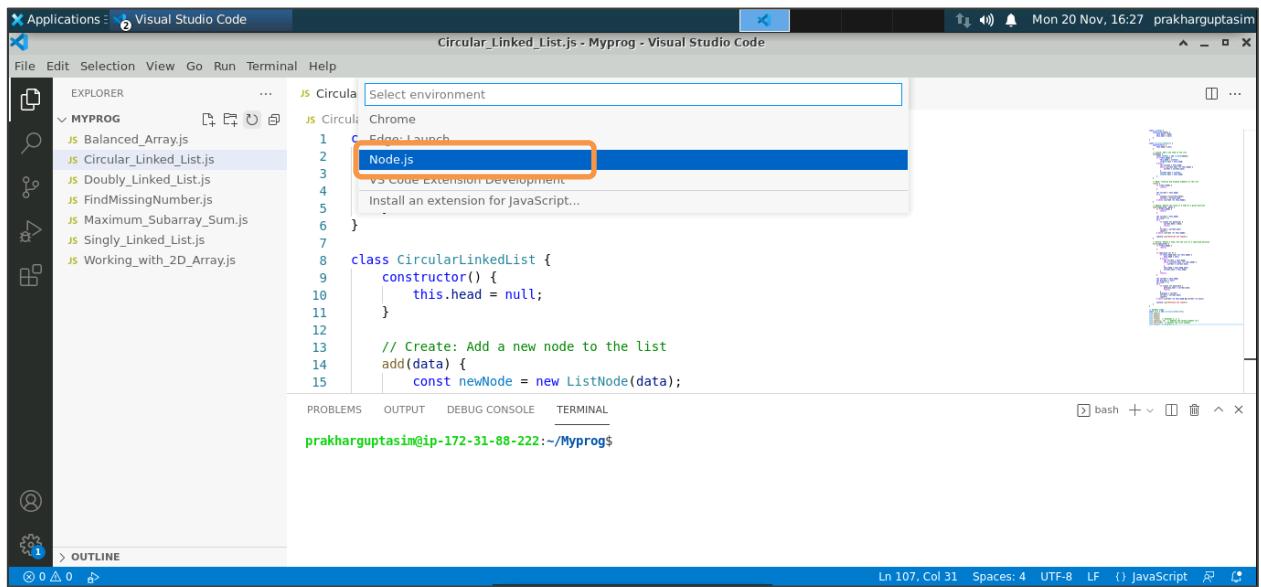
// Example usage
const list = new CircularLinkedList();
list.add(1);
list.add(2);
list.add(3);
list.read(); // Displays 1, 2, 3
list.update(1, 4); // Updates the second element to 4
list.delete(0); // Deletes the first element
list.read(); // Displays 4, 3
```

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

- Title Bar:** Applications > Visual Studio Code
- File Explorer (Left):** Shows a folder named "MYPROG" containing several JavaScript files: Balanced_Array.js, Circular_Linked_List.js (selected), Doubly_Linked_List.js, FindMissingNumber.js, Maximum_Subarray_Sum.js, Singly_Linked_List.js, and Working_with_2D_Array.js.
- Code Editor (Center):** Displays the content of Circular_Linked_List.js. The code defines a ListNode class and a CircularLinkedList class, both used to create a circular linked list.
- Terminal (Bottom):** Shows the command prompt: prakharguptasim@ip-172-31-88-222:~/Myprog\$
- Status Bar (Bottom):** Shows file information: Ln 107, Col 31, Spaces: 4, UTF-8, LF, and a JavaScript icon.
- Notification (Bottom Right):** A message box indicates an available update, with options to "Download Update", "Later", or "Release Notes".

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



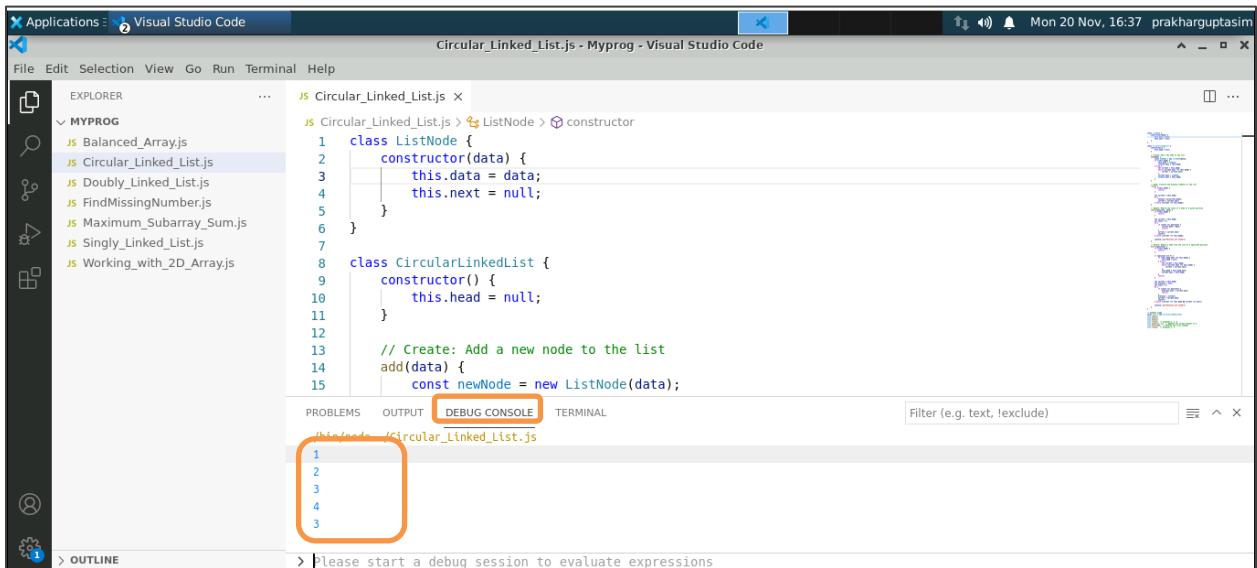


The screenshot shows the Visual Studio Code interface. In the top right corner, the status bar indicates "Mon 20 Nov, 16:27 prakharguptasim". The main area displays a file named "Circular_Linked_List.js" with the following code:

```
JS Circular_Linked_List.js - Myprog - Visual Studio Code
Select environment
JS Circular_Linked_List.js
1 | Circular_Linked_List.js > ListNode > constructor
2 | class ListNode {
3 |   constructor(data) {
4 |     this.data = data;
5 |     this.next = null;
6 |   }
7 |
8 | class CircularLinkedList {
9 |   constructor() {
10 |     this.head = null;
11 |   }
12 |
13 |   // Create: Add a new node to the list
14 |   add(data) {
15 |     const newNode = new ListNode(data);
```

The "DEBUG CONSOLE" tab is selected at the bottom, showing the command "prakharguptasim@ip-172-31-88-222:~/Myprog\$". The status bar at the bottom right shows "Ln 107, Col 31 Spaces: 4 UTF-8 LF () JavaScript".

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



The screenshot shows the Visual Studio Code interface with the "DEBUG CONSOLE" tab selected. The code editor shows the same "Circular_Linked_List.js" file as before. The DEBUG CONSOLE window displays the following output:

```
JS Circular_Linked_List.js - Myprog - Visual Studio Code
Circular_Linked_List.js > ListNode > constructor
1 | Circular_Linked_List.js > ListNode > constructor
2 | class ListNode {
3 |   constructor(data) {
4 |     this.data = data;
5 |     this.next = null;
6 |   }
7 |
8 | class CircularLinkedList {
9 |   constructor() {
10 |     this.head = null;
11 |   }
12 |
13 |   // Create: Add a new node to the list
14 |   add(data) {
15 |     const newNode = new ListNode(data);
```

The DEBUG CONSOLE window also contains the message "please start a debug session to evaluate expressions". The status bar at the bottom right shows "Ln 107, Col 31 Spaces: 4 UTF-8 LF () JavaScript".

By following these steps, you have successfully performed the CRUD operations on a circular linked list, strengthening your understanding of circular data structure operations. In this process, the **add()** method appends a new node at the end of the list, the **read()** method traverses and prints the list, the **update()** method modifies the value at a given position, and the **delete()** method removes a node at a specified position.