

Lesson 02 Demo 04

Implementing CRUD Operations on a Singly Linked List

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

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

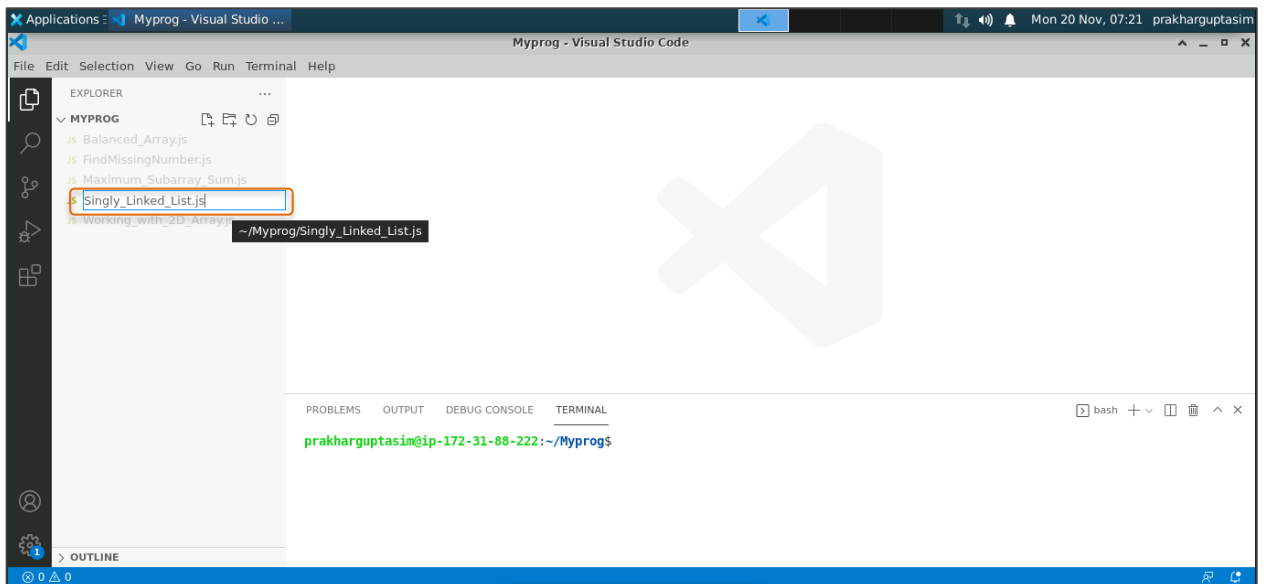
Prerequisites: Completion of Lesson 02 Demo 01

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 **Singly_Linked_List.js**



1.2 Add the following code to the file:

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

class SinglyLinkedList {
  constructor() {
    this.head = 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;
    } else {
      let current = this.head;
      while (current.next) {
        current = current.next;
      }
      current.next = newNode;
    }
  }

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

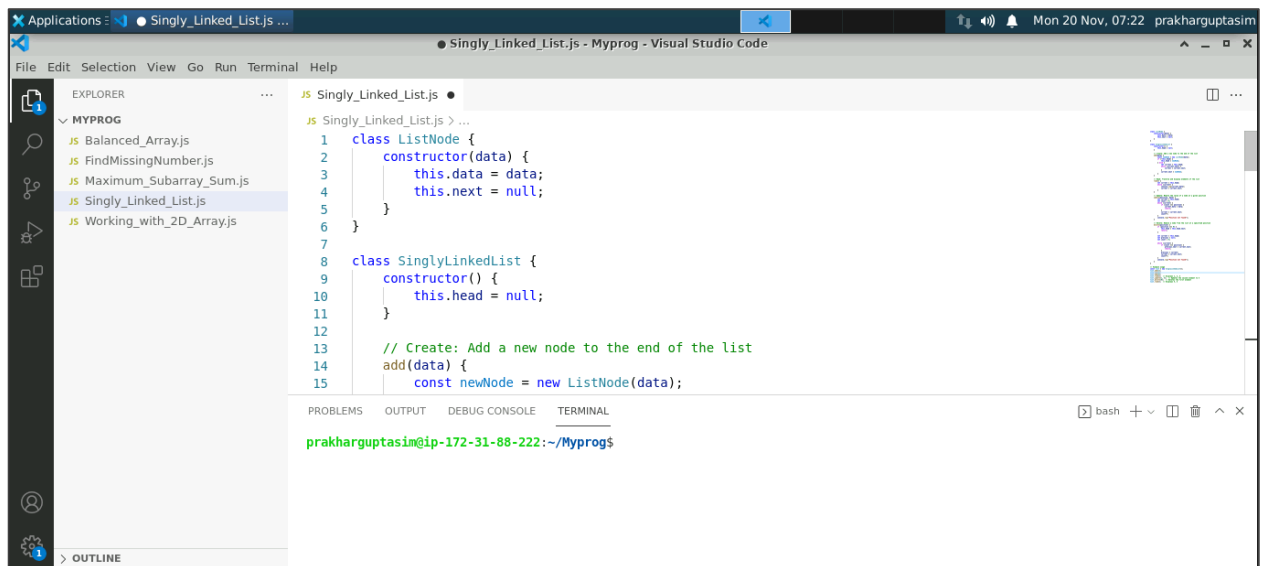
// Update: Modify the value of a node at a given position

```
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;  
    return;  
  }  
  
  let current = this.head;  
  let previous = null;  
  let count = 0;  
  
  while (current) {  
    if (count === position) {  
      previous.next = current.next;  
      return;  
    }  
    previous = current;  
    current = current.next;  
    count++;  
  }  
  console.log("Position not found");  
}}
```

```
// Example usage
const list = new SinglyLinkedList();
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
```

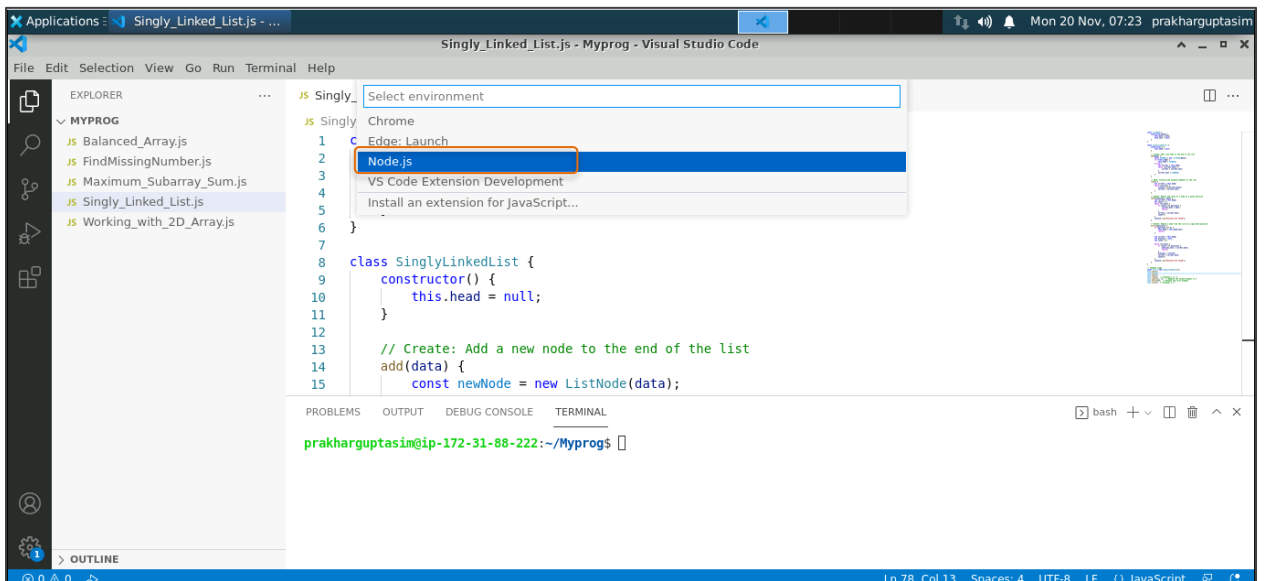
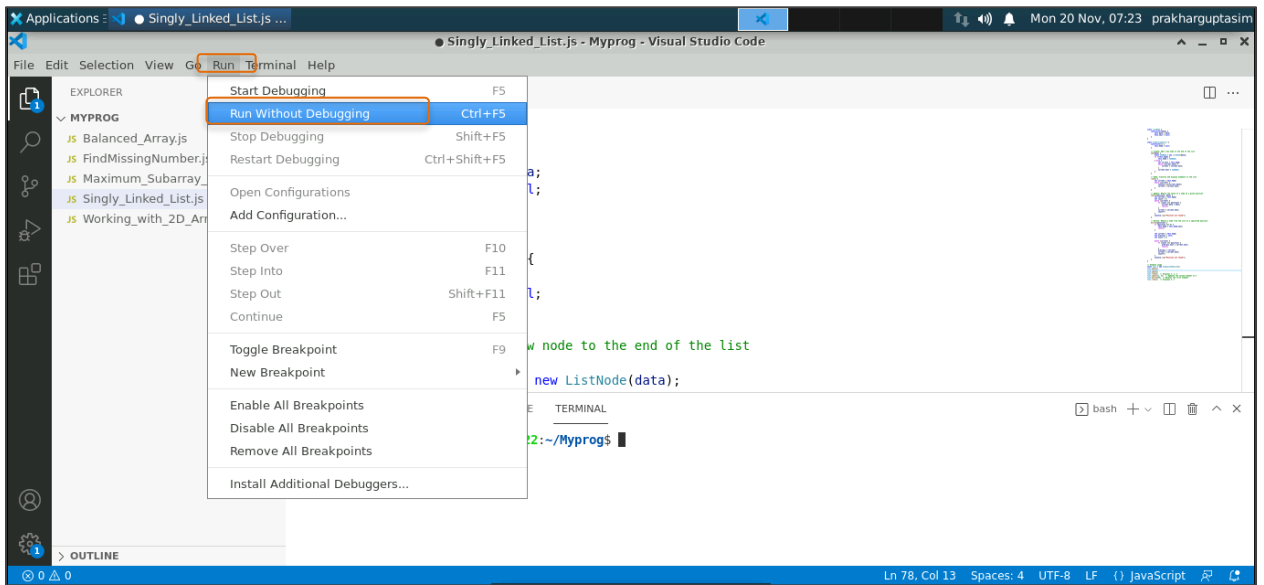


The screenshot shows a Visual Studio Code editor window with the file `Singly_Linked_List.js` open. The code defines a `SinglyLinkedList` class with the following methods:

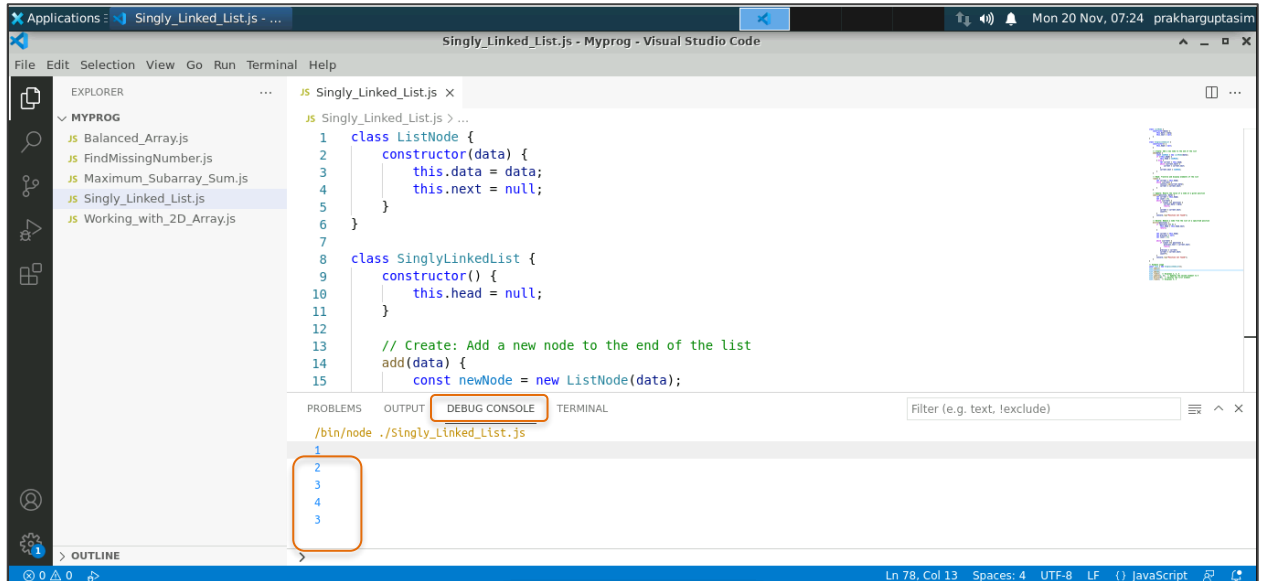
- `constructor()`: Initializes the `head` property to `null`.
- `add(data)`: Adds a new node to the end of the list.
- `read()`: Displays the elements of the list.
- `update(index, data)`: Updates the element at the specified index.
- `delete(index)`: Deletes the element at the specified index.

The terminal at the bottom shows the command prompt for the user `prakharguptasim` on a machine with IP `172-31-88-222`.

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 a file named `Singly_Linked_List.js` open. The Explorer sidebar on the left shows a folder named `MYPROG` containing several JavaScript files, with `Singly_Linked_List.js` selected. The main editor displays the following JavaScript code:

```
1 class ListNode {
2   constructor(data) {
3     this.data = data;
4     this.next = null;
5   }
6 }
7
8 class SinglyLinkedList {
9   constructor() {
10    this.head = null;
11  }
12
13  // Create: Add a new node to the end of the List
14  add(data) {
15    const newNode = new ListNode(data);
```

Below the editor, the **DEBUG CONSOLE** tab is active, showing the command `./bin/node ./Singly_Linked_List.js` and its output:

```
1
2
3
4
5
```

The output shows five lines, with the last line being a blank space. The first four lines are numbered 1 through 4, and the fifth line is numbered 5. The numbers 1, 2, 3, 4, and 5 are highlighted with a red box.

By following these steps, you have successfully performed CRUD operations on a singly linked list. The **add()** method adds a new node at the end of the list, the **read()** method traverses and prints the list, the **update()** method changes the value at a given position, and the **delete()** method removes a node at a specified position. These are essential operations when working with dynamic data structures.