Lesson 02 Demo 13

Implementing Stacks Using a Linked List

Objective: To demonstrate the implementation of a stack using a linked list in JavaScript, covering operations like push, pop, peek, display, and clear

Tools required: Visual Studio Code and Node.js

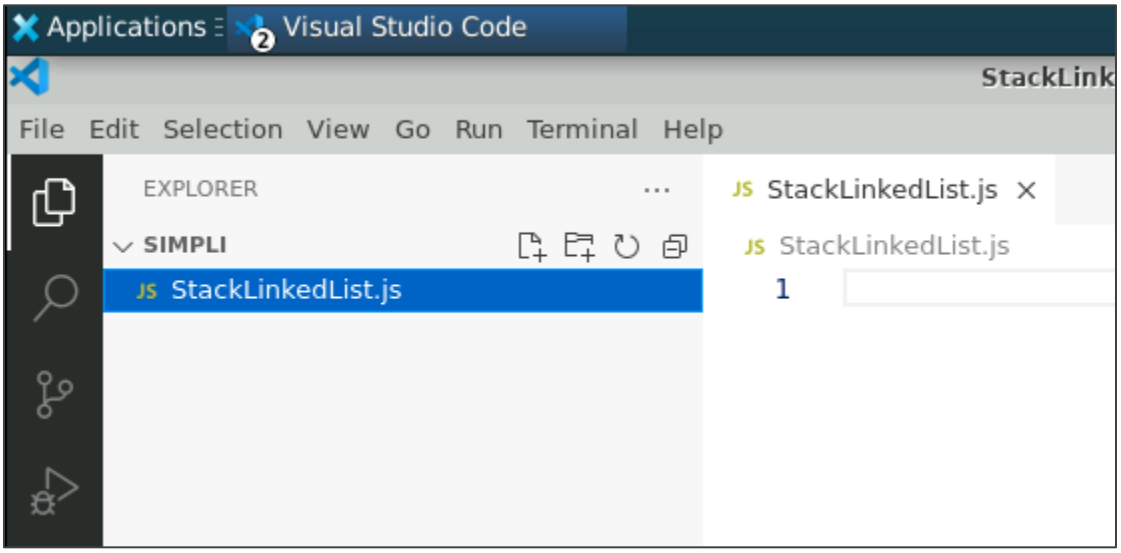
Prerequisites: Basic understanding of linked lists in JavaScript

Steps to be followed:

1. Create and execute the JS file

Step 1: Create and execute the JS file

1.1 Open the Visual Studio Code editor and create a JavaScript file named **StackLinkedList.js**



1.2 Write the code given below in the **StackLinkedList.js** file:

```
// Implementing a Stack using Linked List

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

class StackLinkedList {
  constructor() {
    this.top = null;
    this.size = 0;
  }

  // Push operation
  push(element) {
    const newNode = new Node(element);
    newNode.next = this.top;
    this.top = newNode;
    this.size++;
  }

  // Pop operation
  pop() {
    if (this.size === 0) {
      return "Underflow";
    }
    const poppedNode = this.top;
    this.top = this.top.next;
    this.size--;
    return poppedNode.data;
  }

  // Peek operation
  peek() {
    return this.top ? this.top.data : null;
  }
```

```
    // Display the Stack
    display() {
        let current = this.top;
        while (current) {
            console.log(current.data);
            current = current.next;
        }
    }

    // Clear the Stack
    clear() {
        this.top = null;
        this.size = 0;
        console.log("Stack cleared.");
    }
}

// Creating a stack
let myStack = new StackLinkedList();

// Pushing elements onto the stack
myStack.push(10);
myStack.push(20);
myStack.push(30);

// Displaying the stack
myStack.display();

// Popping an element from the stack
let poppedElement = myStack.pop();
console.log("Popped element:", poppedElement);

// Displaying the stack after popping
myStack.display();

// Peeking into the stack
let topElement = myStack.peek();
console.log("Top element:", topElement);

// Clearing the stack
myStack.clear();
myStack.display();
```

JS StackLinkedList.js > ...

```
1   // Implementing a Stack using Linked List
2
3   class Node {
4       constructor(data) {
5           this.data = data;
6           this.next = null;
7       }
8   }
9
10  class StackLinkedList {
11      constructor() {
12          this.top = null;
13          this.size = 0;
14      }
15
16      // Push operation
17      push(element) {
18          const newNode = new Node(element);
19          newNode.next = this.top;
20          this.top = newNode;
21          this.size++;
22      }
23
```

```
24      // Pop operation
25      pop() {
26          if (this.size === 0) {
27              return "Underflow";
28          }
29          const poppedNode = this.top;
30          this.top = this.top.next;
31          this.size--;
32          return poppedNode.data;
33      }
34
35      // Peek operation
36      peek() {
37          return this.top ? this.top.data : null;
38      }
39
```

```
40      // Display the Stack
41      display() {
42          let current = this.top;
43          while (current) {
44              console.log(current.data);
45              current = current.next;
46          }
47      }
48
49      // Clear the Stack
50      clear() {
51          this.top = null;
52          this.size = 0;
53          console.log("Stack cleared.");
54      }
55  }
56
57  // Creating a stack
58  let myStack = new StackLinkedList();
59
```

```
60  // Pushing elements onto the stack
61  myStack.push(10);
62  myStack.push(20);
63  myStack.push(30);
64
65  // Displaying the stack
66  myStack.display();
67
68  // Popping an element from the stack
69  let poppedElement = myStack.pop();
70  console.log("Popped element:", poppedElement);
71
72  // Displaying the stack after popping
73  myStack.display();
74
75  // Peeking into the stack
76  let topElement = myStack.peek();
77  console.log("Top element:", topElement);
78
79  // Clearing the stack
80  myStack.clear();
81  myStack.display();
```

1.3 Save the file and execute it in the terminal using the command given below:
StackLinkedList.js

```
65   // Displaying the stack
66   myStack.display();
67
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

priyanshurajsim@ip-172-31-29-42:~/Downloads/Simpli$ ls
StackLinkedList.js
priyanshurajsim@ip-172-31-29-42:~/Downloads/Simpli$ node StackLinkedList.js
30
20
10
Popped element: 30
20
10
Top element: 20
Stack cleared.
priyanshurajsim@ip-172-31-29-42:~/Downloads/Simpli$
```

This example demonstrates the implementation of a stack using a linked list, including pushing, popping, peeking, displaying, and clearing elements.

By following these steps, you have successfully mastered the implementation and manipulation of a stack using a linked list in JavaScript, gaining valuable insights into dynamic data structure operations.