Lesson 02 Demo 11

Implementing CRUD Operations on a Stack

Objective: To implement CRUD operations on a stack, showcasing how to push, pop, peek, display, and clear elements within a stack structure

Tools required: Visual Studio Code and Node.js

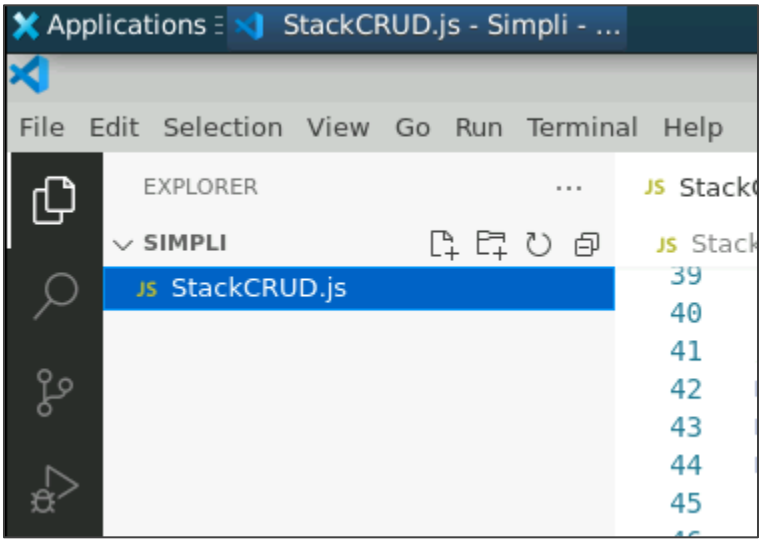
Prerequisites: Basic understanding of data structures and JavaScript

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 **StackCRUD.js**



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

```
// Implementing CRUD Operations on a Stack

class Stack {
  constructor() {
    this.items = [];
  }

  // Push operation
  push(element) {
    this.items.push(element);
  }

  // Pop operation
  pop() {
    if (this.items.length === 0) {
      return "Underflow";
    }
    return this.items.pop();
  }

  // Peek operation
  peek() {
    return this.items[this.items.length - 1];
  }

  // Display the Stack
  display() {
    console.log("Stack elements:", this.items);
  }

  // Clear the Stack
  clear() {
    this.items = [];
    console.log("Stack cleared.");
  }
}

// Creating a stack
let myStack = new Stack();
```

```
// 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 StackCRUD.js > Stack > constructor

```
// Implementing CRUD Operations on a Stack

class Stack {
    constructor() {
        this.items = [];
    }

    // Push operation
    push(element) {
        this.items.push(element);
    }

    // Pop operation
    pop() {
        if (this.items.length === 0) {
            return "Underflow";
        }
        return this.items.pop();
    }

    // Peek operation
    peek() {
        return this.items[this.items.length - 1];
    }

    // Display the Stack
    display() {
        console.log("Stack elements:", this.items);
    }

    // Clear the Stack
    clear() {
        this.items = [];
        console.log("Stack cleared.");
    }
}

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

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

```
46  // Displaying the stack
47  myStack.display();
48
49  // Popping an element from the stack
50  let poppedElement = myStack.pop();
51  console.log("Popped element:", poppedElement);
52
53  // Displaying the stack after popping
54  myStack.display();
55
56  // Peeking into the stack
57  let topElement = myStack.peek();
58  console.log("Top element:", topElement);
59
60  // Clearing the stack
61  myStack.clear();
62  myStack.display();
```

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

```
49  // Popping an element from the stack

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

priyanshurajsim@ip-172-31-39-132:~/Downloads/Simpli$ ls
StackCRUD.js
priyanshurajsim@ip-172-31-39-132:~/Downloads/Simpli$ node StackCRUD.js
Stack elements: [ 10, 20, 30 ]
Popped element: 30
Stack elements: [ 10, 20 ]
Top element: 20
Stack cleared.
Stack elements: []
priyanshurajsim@ip-172-31-39-132:~/Downloads/Simpli$
```

This example illustrates CRUD operations on a stack, including pushing, popping, peeking, displaying, and clearing elements.

Conclusion

By following these steps, you have successfully managed a stack in JavaScript through various CRUD operations, enhancing your ability to manipulate data structures and apply fundamental programming concepts in practical scenarios.