

Lesson 02 Demo 15

Implementing Stack and Queue using Deque

Objective: To demonstrate the implementation of both stack and queue functionalities using a deque (double-ended queue) in JavaScript

Tools required: Visual Studio Code and Node.js

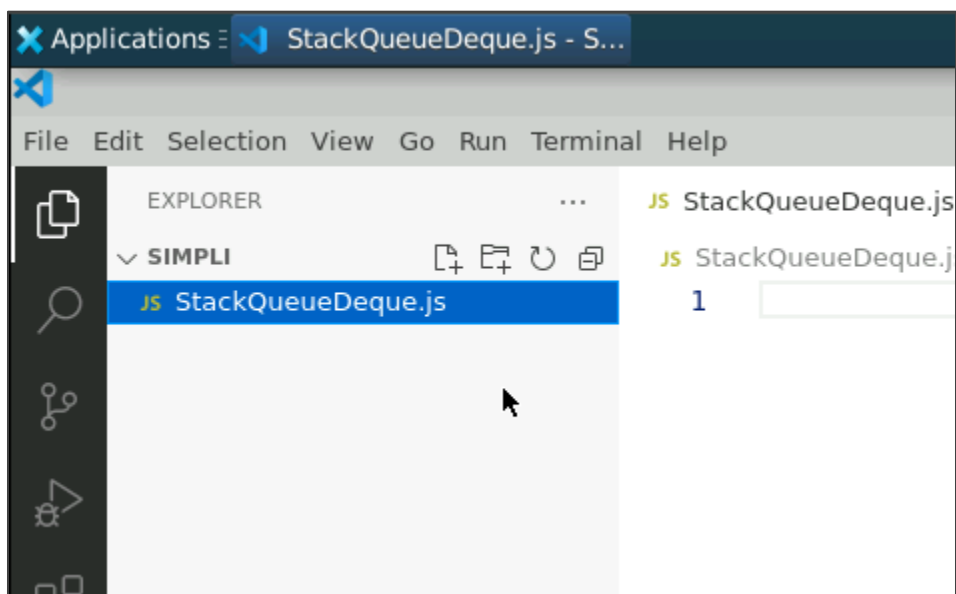
Prerequisites: Basic understanding of stack, queue, deque and JavaScript

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 **StackQueueDeque.js**



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

```
// Deque implementation
class Deque {
  constructor() {
    this.items = [];
  }

  // Methods for Stack implementation
  push(item) {
    this.items.push(item);
  }

  pop() {
    if (this.isEmpty()) {
      return undefined;
    }
    return this.items.pop();
  }

  // Methods for Queue implementation
  enqueue(item) {
    this.items.push(item);
  }

  dequeue() {
    if (this.isEmpty()) {
      return undefined;
    }
    return this.items.shift();
  }

  isEmpty() {
    return this.items.length === 0;
  }

  size() {
    return this.items.length;
  }
}
```

```
// Example usage
const stack = new Deque();
stack.push(1);
stack.push(2);
console.log('Stack pop:', stack.pop());

const queue = new Deque();
queue.enqueue(1);
queue.enqueue(2);
console.log('Queue dequeue:', queue.dequeue());
```

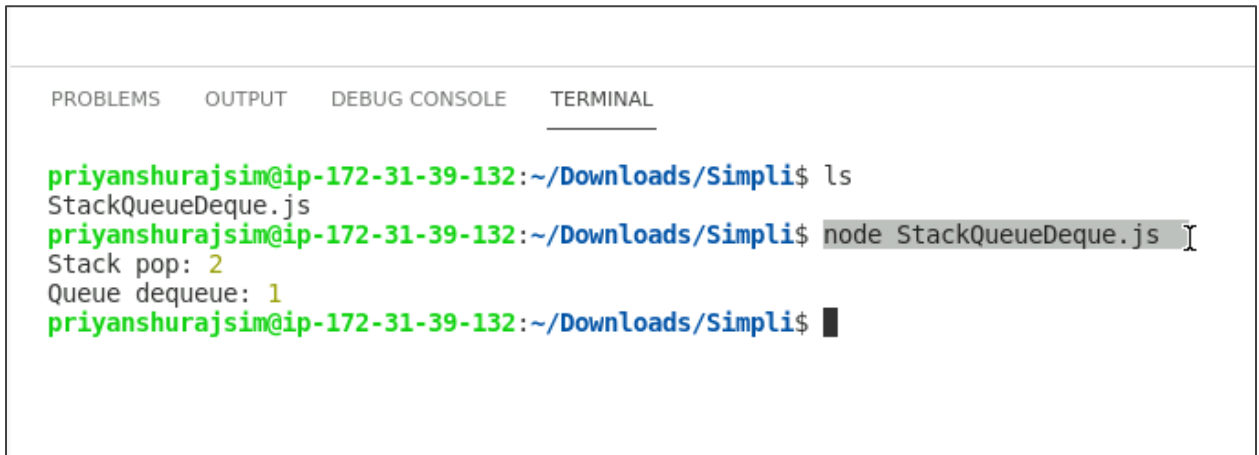
```
1 // Deque implementation
2
3 class Deque {
4   constructor() {
5     this.items = [];
6   }
7
8   // Methods for Stack implementation
9   push(item) {
10    this.items.push(item);
11  }
12
13  pop() {
14    if (this.isEmpty()) {
15      return undefined;
16    }
17    return this.items.pop();
18  }
19 }
```

```
20 // Methods for Queue implementation
21 enqueue(item) {
22     this.items.push(item);
23 }
24
25 dequeue() {
26     if (this.isEmpty()) {
27         return undefined;
28     }
29     return this.items.shift();
30 }
31
32 isEmpty() {
33     return this.items.length === 0;
34 }
35
36 size() {
37     return this.items.length;
38 }
39 }
40
```

```
41 // Example usage
42 const stack = new Deque();
43 stack.push(1);
44 stack.push(2);
45 console.log('Stack pop:', stack.pop());
46
47 const queue = new Deque();
48 queue.enqueue(1);
49 queue.enqueue(2);
50 console.log('Queue dequeue:', queue.dequeue());
51
```

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

node StackQueueDeque.js



The screenshot shows a terminal window with a tab labeled 'TERMINAL'. The prompt is 'priyanshurajsim@ip-172-31-39-132:~/Downloads/Simpli\$'. The first command is 'ls', which lists 'StackQueueDeque.js'. The second command is 'node StackQueueDeque.js', which produces the output 'Stack pop: 2' and 'Queue dequeue: 1'. The prompt returns to 'priyanshurajsim@ip-172-31-39-132:~/Downloads/Simpli\$'.

```
priyanshurajsim@ip-172-31-39-132:~/Downloads/Simpli$ ls
StackQueueDeque.js
priyanshurajsim@ip-172-31-39-132:~/Downloads/Simpli$ node StackQueueDeque.js
Stack pop: 2
Queue dequeue: 1
priyanshurajsim@ip-172-31-39-132:~/Downloads/Simpli$
```

This example illustrates the implementation of stack and queue using a deque in JavaScript.

By following these steps, you have successfully implemented stack and queue operations using a deque in JavaScript, broadening your understanding of versatile data structures and their applications in programming.