

Lesson 02 Demo 13

Implementing CRUD Operations on a Queue

Objective: To implement CRUD operations on a queue in JavaScript, including enqueue, dequeue, accessing elements, and checking the queue's size and emptiness, enhancing your ability to manage and interact effectively with fundamental data structures

Tools required: Visual Studio Code and Node.js

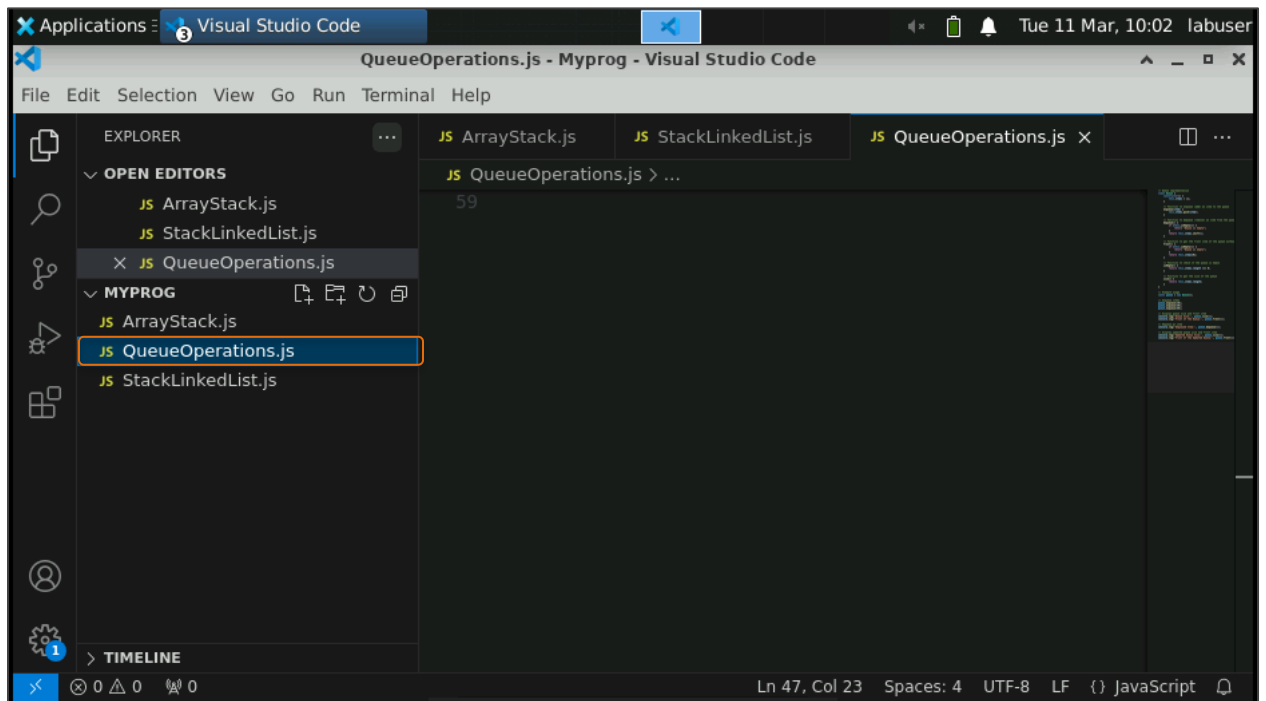
Prerequisites: A basic understanding of queues and JavaScript

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 **QueueOperations.js**



1.2 Add the following code to the file:

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

  // Function to enqueue (add) an item to the queue
  enqueue(item) {
    this.items.push(item);
  }

  // Function to dequeue (remove) an item from the queue
  dequeue() {
    if (this.isEmpty()) {
      return "Queue is empty";
    }
    return this.items.shift();
  }

  // Function to get the front item of the queue without removing it
  front() {
    if (this.isEmpty()) {
      return "Queue is empty";
    }
    return this.items[0];
  }

  // Function to check if the queue is empty
  isEmpty() {
    return this.items.length === 0;
  }

  // Function to get the size of the queue
  size() {
    return this.items.length;
  }
}

// Example usage
const queue = new Queue();

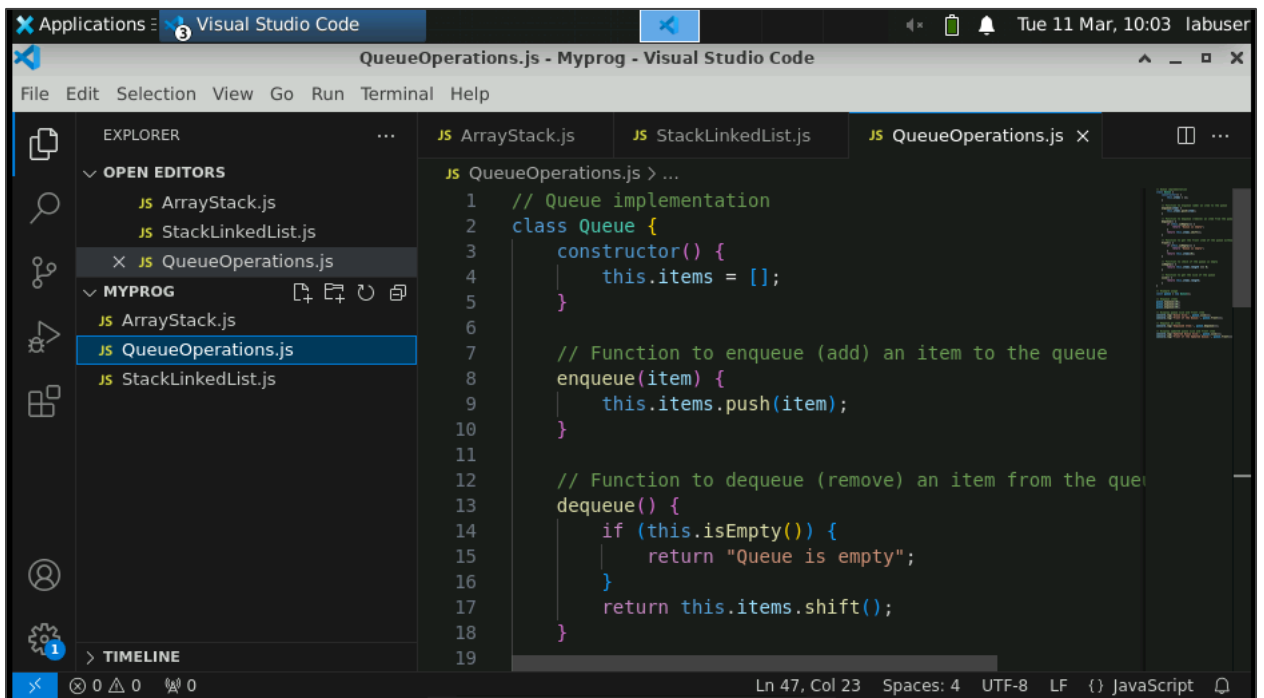
// Enqueue items
```

```
queue.enqueue(10);
queue.enqueue(20);
queue.enqueue(30);
```

```
// Display queue size and front item
console.log('Queue Size:', queue.size());
console.log('Front of the Queue:', queue.front());
```

```
// Dequeue an item
console.log('Dequeued Item:', queue.dequeue());
```

```
// Display updated queue size and front item
console.log('Updated Queue Size:', queue.size());
console.log('Front of the Updated Queue:', queue.front());
```



```
QueueOperations.js - Myprog - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
  OPEN EDITORS
    JS ArrayStack.js
    JS StackLinkedList.js
    JS QueueOperations.js
  MYPROG
    JS ArrayStack.js
    JS QueueOperations.js
    JS StackLinkedList.js

TIMELINE

JS QueueOperations.js > ...
1  // Queue implementation
2  class Queue {
3      constructor() {
4          this.items = [];
5      }
6
7      // Function to enqueue (add) an item to the queue
8      enqueue(item) {
9          this.items.push(item);
10     }
11
12     // Function to dequeue (remove) an item from the queue
13     dequeue() {
14         if (this.isEmpty()) {
15             return "Queue is empty";
16         }
17         return this.items.shift();
18     }
19 }
```

Visual Studio Code interface showing the `QueueOperations.js` file. The Explorer sidebar shows the project structure with `ArrayStack.js`, `StackLinkedList.js`, and `QueueOperations.js`. The main editor displays the following code:

```
2 class Queue {
20     // Function to get the front item of the queue without
21     front() {
22         if (this.isEmpty()) {
23             return "Queue is empty";
24         }
25         return this.items[0];
26     }
27
28     // Function to check if the queue is empty
29     isEmpty() {
30         return this.items.length === 0;
31     }
32
33     // Function to get the size of the queue
34     size() {
35         return this.items.length;
36     }
37 }
```

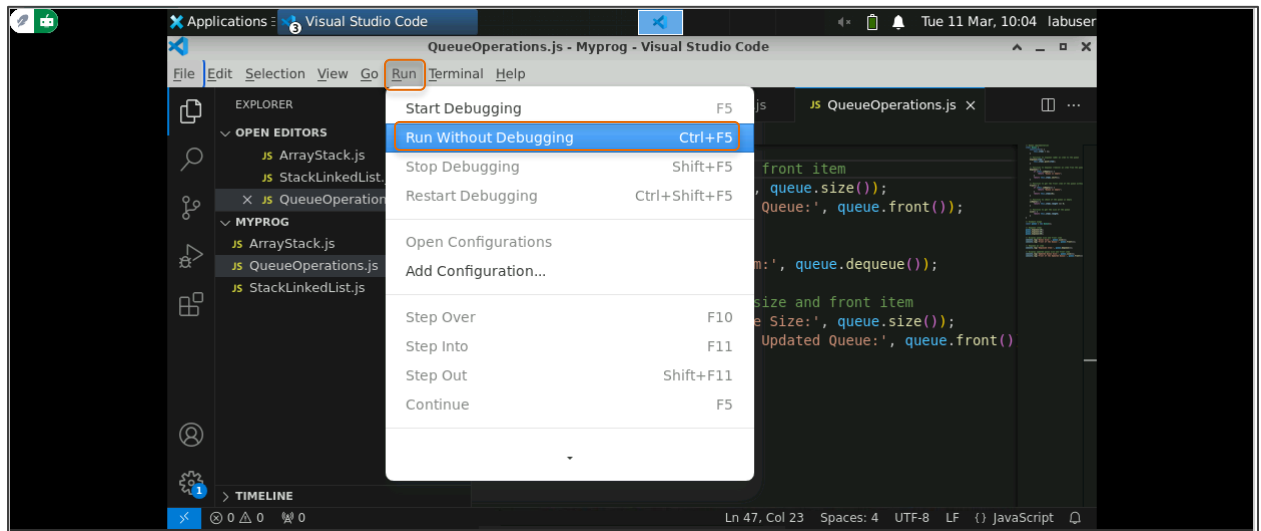
The status bar at the bottom indicates the cursor is at `Ln 47, Col 23`.

Visual Studio Code interface showing the `QueueOperations.js` file. The Explorer sidebar shows the project structure with `ArrayStack.js`, `StackLinkedList.js`, and `QueueOperations.js`. The main editor displays the following code:

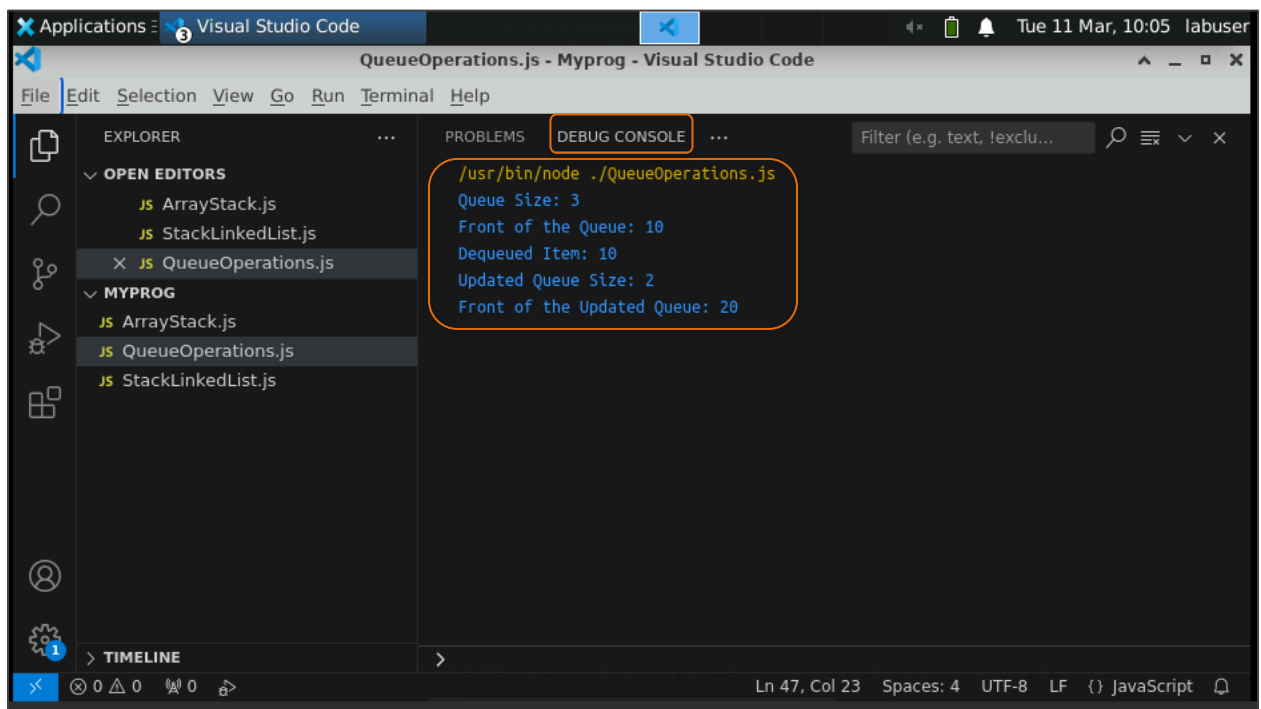
```
39 // Example usage
40 const queue = new Queue();
41
42 // Enqueue items
43 queue.enqueue(10);
44 queue.enqueue(20);
45 queue.enqueue(30);
46
47 // Display queue size and front item
48 console.log('Queue Size:', queue.size());
49 console.log('Front of the Queue:', queue.front());
50
51 // Dequeue an item
52 console.log('Dequeued Item:', queue.dequeue());
53
54 // Display updated queue size and front item
55 console.log('Updated Queue Size:', queue.size());
56 console.log('Front of the Updated Queue:', queue.front());
57
```

The status bar at the bottom indicates the cursor is at `Ln 47, Col 23`.

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



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



Note: This example illustrates CRUD operations on a queue using JavaScript.

By following these steps, you have successfully implemented and executed CRUD operations on a queue in JavaScript, enhancing your ability to manage and interact effectively with fundamental data structures.