

Lesson 03 Demo 04

Creating and Representing Graph

Objective: To demonstrate the creation and representation of a graph using JavaScript

Tools required: Visual Studio Code and Node.js

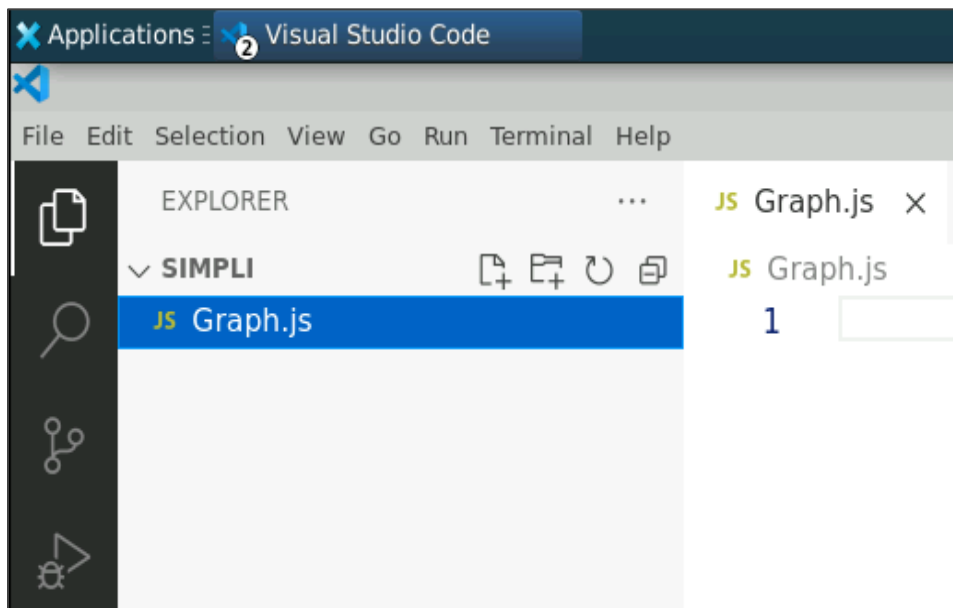
Prerequisites: Basic understanding of data structures and JavaScript

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 **Graph.js**



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

```
// Graph implementation using adjacency list
class Graph {
  constructor() {
    this.vertices = [];
    this.adjacencyList = new Map();
  }

  // Function to add a vertex to the graph
  addVertex(vertex) {
    this.vertices.push(vertex);
    this.adjacencyList.set(vertex, []);
  }

  // Function to add an edge between two vertices
  addEdge(vertex1, vertex2) {
    this.adjacencyList.get(vertex1).push(vertex2);
    this.adjacencyList.get(vertex2).push(vertex1);
  }

  // Function to display the graph
  printGraph() {
    for (const vertex of this.vertices) {
      const neighbors = this.adjacencyList.get(vertex).join(' ');
      console.log(`${vertex} -> ${neighbors}`);
    }
  }
}

// Example usage
const graph = new Graph();
graph.addVertex('A');
graph.addVertex('B');
graph.addVertex('C');
graph.addEdge('A', 'B');
graph.addEdge('B', 'C');

console.log('Graph representation:');
graph.printGraph();
```

JS Graph.js > ...

```
1  // Graph implementation using adjacency list
2  class Graph {
3      constructor() {
4          this.vertices = [];
5          this.adjacencyList = new Map();
6      }
7
8      // Function to add a vertex to the graph
9      addVertex(vertex) {
10         this.vertices.push(vertex);
11         this.adjacencyList.set(vertex, []);
12     }
13
14     // Function to add an edge between two vertices
15     addEdge(vertex1, vertex2) {
16         this.adjacencyList.get(vertex1).push(vertex2);
17         this.adjacencyList.get(vertex2).push(vertex1);
18     }
19 
```

```
20     // Function to display the graph
21     printGraph() {
22         for (const vertex of this.vertices) {
23             const neighbors = this.adjacencyList.get(vertex).join(', ');
24             console.log(`${vertex} -> ${neighbors}`);
25         }
26     }
27 }
28 
```

```
29 // Example usage
30 const graph = new Graph();
31 graph.addVertex('A');
32 graph.addVertex('B');
33 graph.addVertex('C');
34 graph.addEdge('A', 'B');
35 graph.addEdge('B', 'C');
36
37 console.log('Graph representation:');
38 graph.printGraph();
39 
```

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

node Graph.js

```
29 // Example usage
30 const graph = new Graph();
31 graph.addVertex('A');
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
priyanshurajsim@ip-172-31-35-72:~/Downloads/Simpli$ ls
Graph.js
priyanshurajsim@ip-172-31-35-72:~/Downloads/Simpli$ node Graph.js
Graph representation:
A -> B
B -> A, C
C -> B
priyanshurajsim@ip-172-31-35-72:~/Downloads/Simpli$
```

This example demonstrates the creation and representation of a graph in JavaScript using an adjacency list.

By following these steps, you have successfully implemented and executed the process of creating and representing a graph.