Lesson 03 Demo 04

Creating and Representing a Graph

Objective: To demonstrate graph creation and representation in JavaScript using an adjacency list by adding vertices, connecting edges, and displaying the graph structure for clear visualization of relationships

Tools required: Visual Studio Code and Node.js

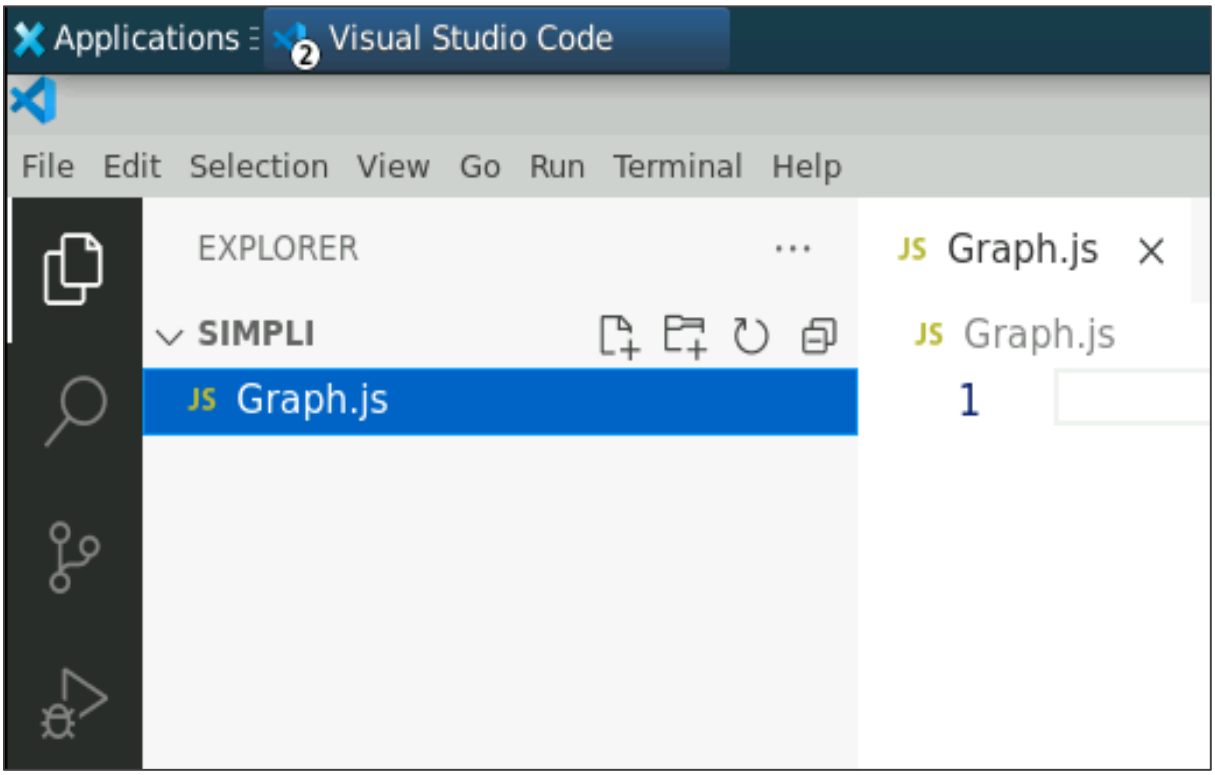
Prerequisites: A basic understanding of data structures and JavaScript

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 **Graph.js**



1.2 Add the following code to the 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 > ...

```
// 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}`);
        }
    }
}
```

```
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 Press **Ctrl** + **S** to save the file and execute it in the **TERMINAL** using the commands given below:

ls

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$
```

By completing these steps, you have successfully created and represented a graph in JavaScript using an adjacency list to visualize relationships clearly.