Lesson 03 Demo 01

Building and Traversing Binary Tree

Objective: To demonstrate the creation and traversal of a binary tree using JavaScript

Tools required: Visual Studio Code and Node.js

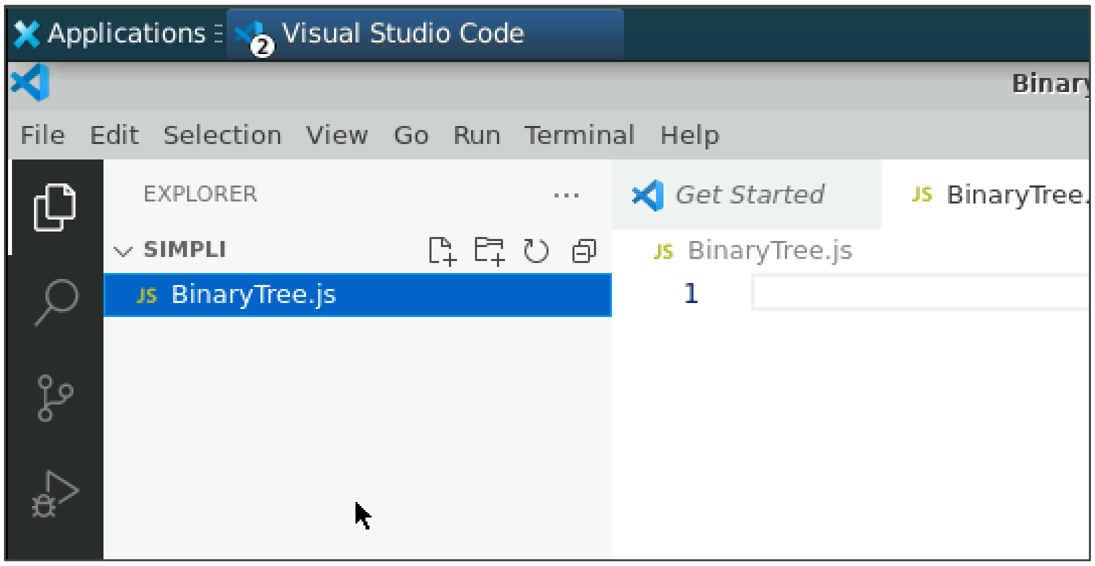
Prerequisites: Basic understanding of data structures and JavaScript

Steps to be followed:

1. Create and execute JS the file

Step 1: Create and execute JS the file

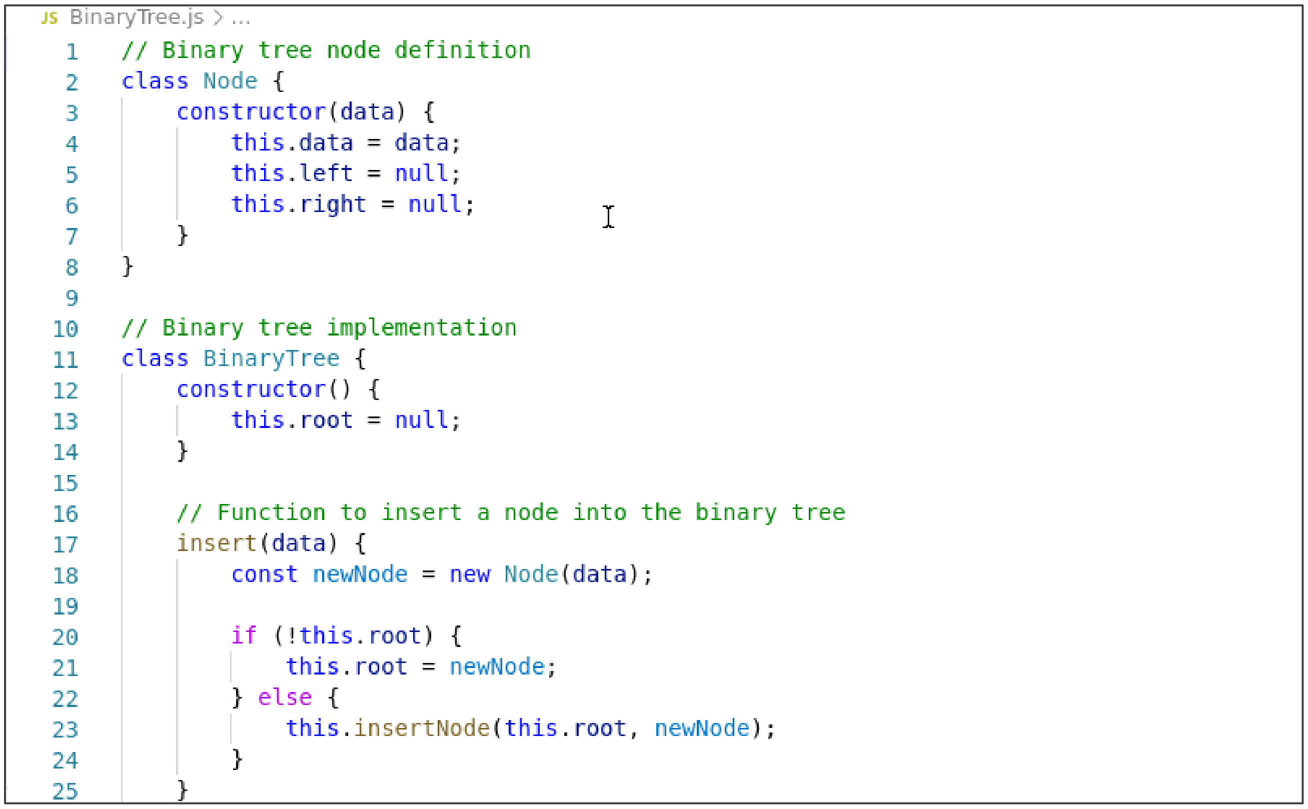
1.1 Open the Visual Studio Code editor and create a JavaScript file named **BinaryTree.js**



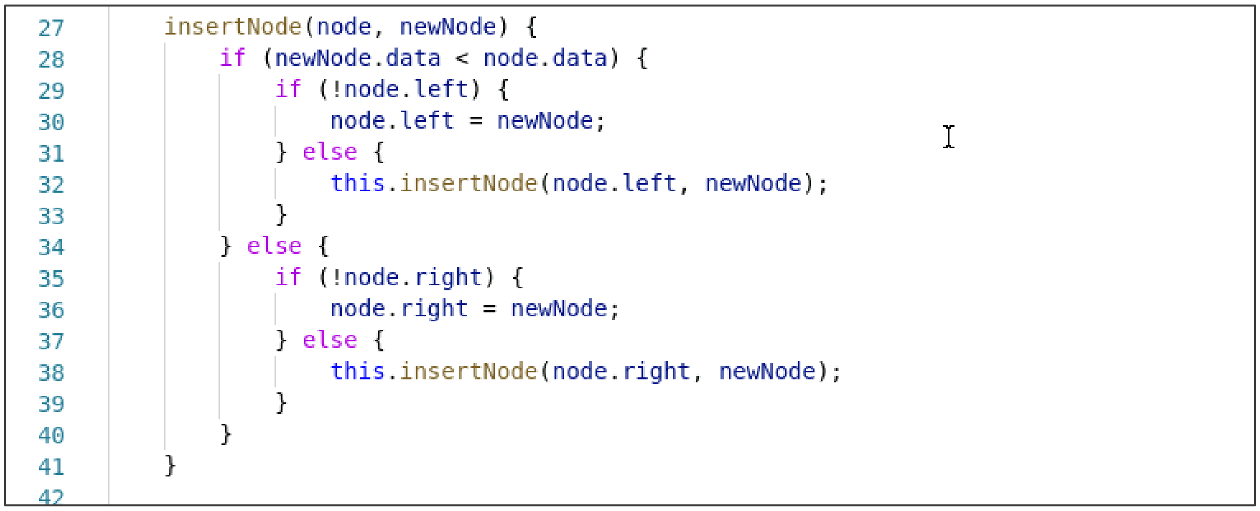
1.2 Write the code given below in the **BinaryTree.js** file:

```
// Binary tree node definition
class Node {
  constructor(data) {
    this.data = data;
    this.left = null;
    this.right = null;
  }
}

// Binary tree implementation
class BinaryTree {
  constructor() {
    this.root = null;
  }

  // Function to insert a node into the binary tree
  insert(data) {
    const newNode = new Node(data);

    if (!this.root) {
      this.root = newNode;
    } else {
      this.insertNode(this.root, newNode);
    }
  }

  insertNode(node, newNode) {
    if (newNode.data < node.data) {
      if (!node.left) {
        node.left = newNode;
      } else {
        this.insertNode(node.left, newNode);
      }
    } else {
      if (!node.right) {
        node.right = newNode;
      } else {
        this.insertNode(node.right, newNode);
      }
    }
  }
```

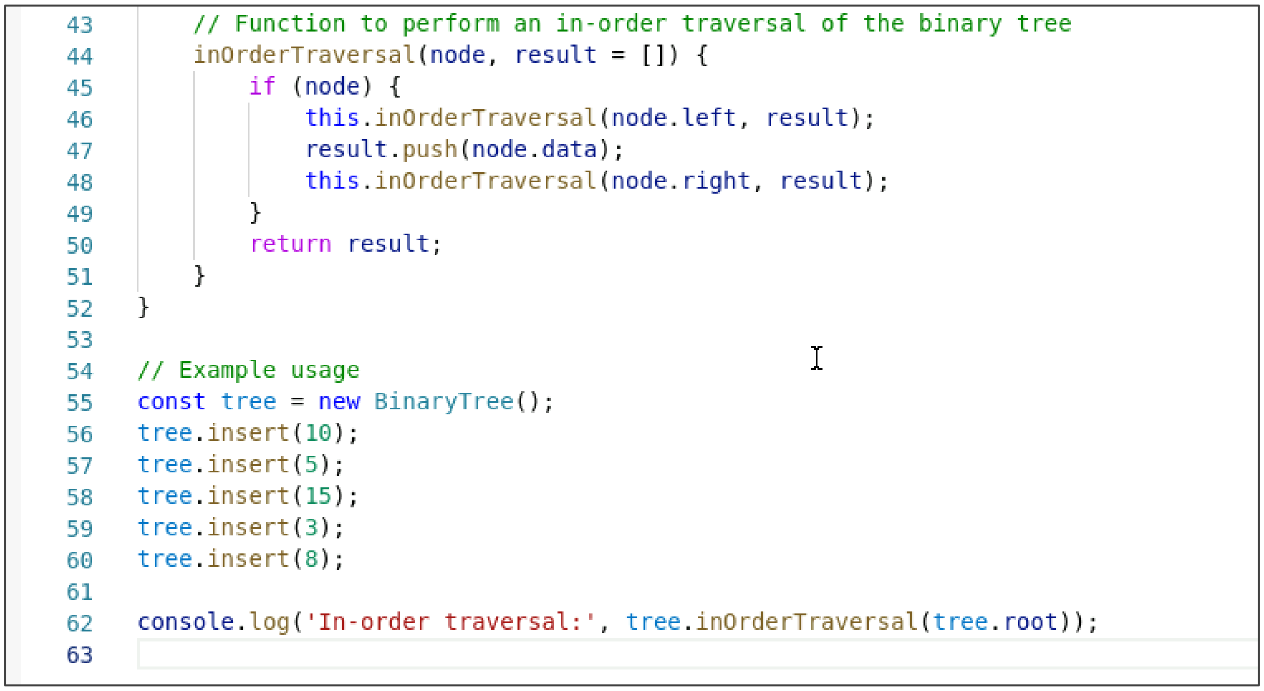
```
    // Function to perform an in-order traversal of the binary tree
    inOrderTraversal(node, result = []) {
      if (node) {
        this.inOrderTraversal(node.left, result);
        result.push(node.data);
        this.inOrderTraversal(node.right, result);
      }
      return result;
    }
}

// Example usage
const tree = new BinaryTree();
tree.insert(10);
tree.insert(5);
tree.insert(15);
tree.insert(3);
tree.insert(8);

console.log('In-order traversal:', tree.inOrderTraversal(tree.root));
```



```
// Binary tree node definition
class Node {
    constructor(data) {
        this.data = data;
        this.left = null;
        this.right = null;
    }
}

// Binary tree implementation
class BinaryTree {
    constructor() {
        this.root = null;
    }

    // Function to insert a node into the binary tree
    insert(data) {
        const newNode = new Node(data);

        if (!this.root) {
            this.root = newNode;
        } else {
            this.insertNode(this.root, newNode);
        }
    }
```



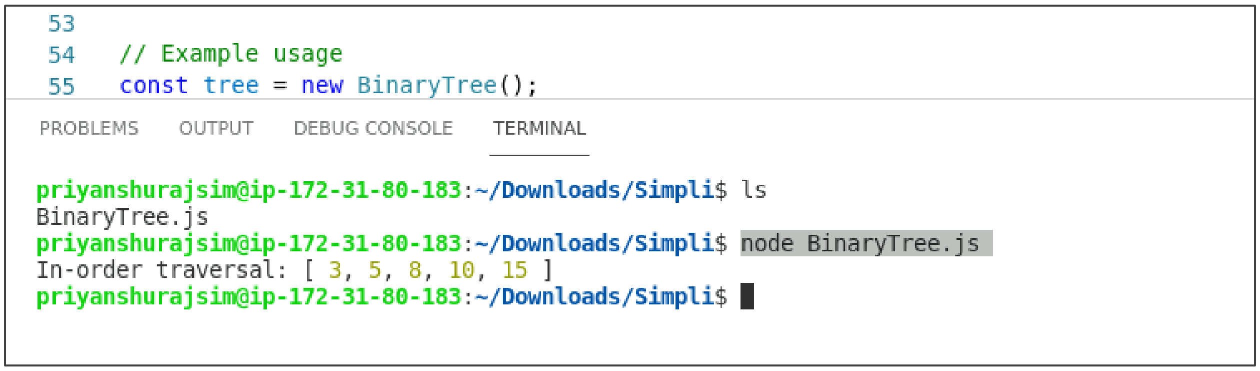
```
    insertNode(node, newNode) {
        if (newNode.data < node.data) {
            if (!node.left) {
                node.left = newNode;
            } else {
                this.insertNode(node.left, newNode);
            }
        } else {
            if (!node.right) {
                node.right = newNode;
            } else {
                this.insertNode(node.right, newNode);
            }
        }
    }

```



```
    // Function to perform an in-order traversal of the binary tree
    inOrderTraversal(node, result = []) {
        if (node) {
            this.inOrderTraversal(node.left, result);
            result.push(node.data);
            this.inOrderTraversal(node.right, result);
        }
        return result;
    }
}

// Example usage
const tree = new BinaryTree();
tree.insert(10);
tree.insert(5);
tree.insert(15);
tree.insert(3);
tree.insert(8);

console.log('In-order traversal:', tree.inOrderTraversal(tree.root));

```

.

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



```
// Example usage
const tree = new BinaryTree();
```

```
priyanshurajsim@ip-172-31-80-183:~/Downloads/Simpli$ ls
BinaryTree.js
priyanshurajsim@ip-172-31-80-183:~/Downloads/Simpli$ node BinaryTree.js
In-order traversal: [ 3, 5, 8, 10, 15 ]
priyanshurajsim@ip-172-31-80-183:~/Downloads/Simpli$
```

This example shows how to build a binary tree and conduct an in-order traversal.

By following these steps, you have successfully implemented and executed the creation and traversal of a binary tree in JavaScript.