

Lesson 03 Demo 01

Building and Traversing a Binary Tree

Objective: To demonstrate binary tree creation and traversal using JavaScript in the context of learning data structures, showcasing how insertion and in-order traversal work through interactive execution

Tools required: Visual Studio Code and Node.js

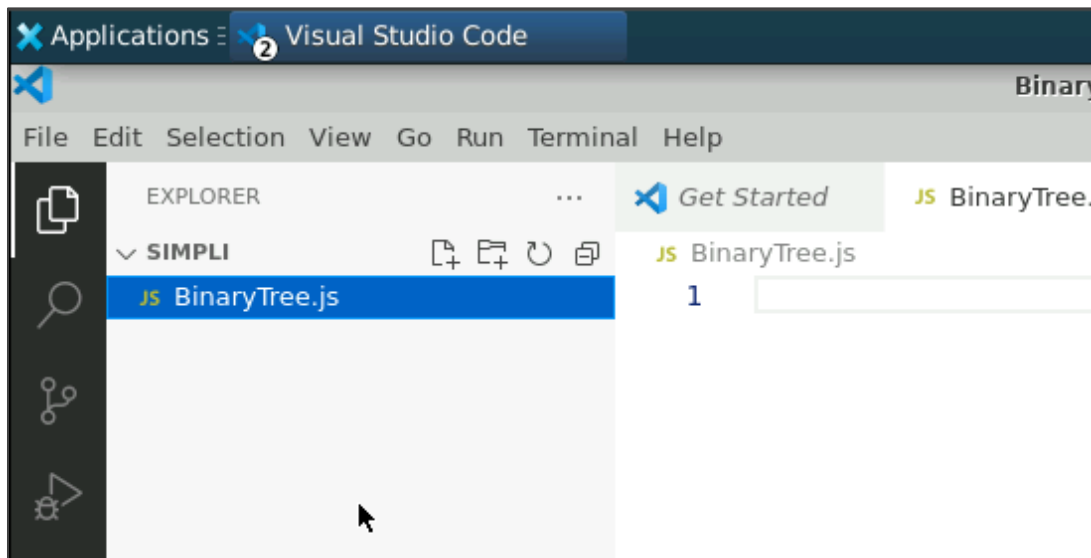
Prerequisites: A basic understanding of data structures and JavaScript

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 **BinaryTree.js**



1.2 Add the following code to 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));

```

```

js BinaryTree.js > ...
1  // Binary tree node definition
2  class Node {
3    constructor(data) {
4      this.data = data;
5      this.left = null;
6      this.right = null;
7    }
8  }
9
10 // Binary tree implementation
11 class BinaryTree {
12   constructor() {
13     this.root = null;
14   }
15
16   // Function to insert a node into the binary tree
17   insert(data) {
18     const newNode = new Node(data);
19
20     if (!this.root) {
21       this.root = newNode;
22     } else {
23       this.insertNode(this.root, newNode);
24     }
25   }

```

```
27     insertNode(node, newNode) {
28         if (newNode.data < node.data) {
29             if (!node.left) {
30                 node.left = newNode;
31             } else {
32                 this.insertNode(node.left, newNode);
33             }
34         } else {
35             if (!node.right) {
36                 node.right = newNode;
37             } else {
38                 this.insertNode(node.right, newNode);
39             }
40         }
41     }
42 }
```

```
43 // Function to perform an in-order traversal of the binary tree
44 inOrderTraversal(node, result = []) {
45     if (node) {
46         this.inOrderTraversal(node.left, result);
47         result.push(node.data);
48         this.inOrderTraversal(node.right, result);
49     }
50     return result;
51 }
52
53
54 // Example usage
55 const tree = new BinaryTree();
56 tree.insert(10);
57 tree.insert(5);
58 tree.insert(15);
59 tree.insert(3);
60 tree.insert(8);
61
62 console.log('In-order traversal:', tree.inOrderTraversal(tree.root));
63
```

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

ls

node BinaryTree.js

```
53
54 // Example usage
55 const tree = new BinaryTree();
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL**

```
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$
```

By following these steps, you have successfully implemented and executed binary tree methods in JavaScript. This example demonstrates key techniques such as searching for a node and finding the smallest value in a binary tree.