

Lesson 03 Demo 02

Working with a Binary Tree

Objective: To demonstrate binary tree operations in JavaScript using node insertion, value searching, and minimum value retrieval to illustrate core algorithmic behavior in tree data structures

Tools required: Visual Studio Code and Node.js

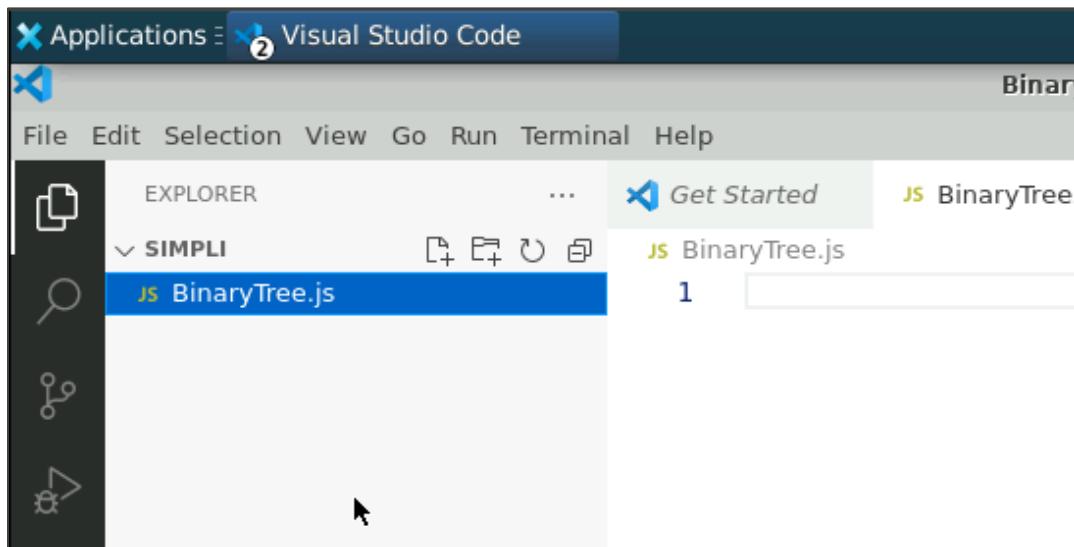
Prerequisites: Familiarity with binary tree basics and JavaScript

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 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) {
        this.root = this._insert(this.root, data);
    }

    _insert(node, data) {
        if (!node) {
            return new Node(data);
        }
        if (data < node.data) {
            node.left = this._insert(node.left, data);
        } else if (data > node.data) {
            node.right = this._insert(node.right, data);
        }
        return node;
    }

    // Function to search for a node in the binary tree
    search(data, node = this.root) {
        if (!node) {
            return false;
        }
        if (data === node.data) {
            return true;
        } else if (data < node.data) {
            return this.search(data, node.left);
        } else {
            return this.search(data, node.right);
        }
    }
}
```

```

// Function to find the minimum value in the binary tree
findMin(node = this.root) {
    if (!node) {
        return null;
    }

    while (node.left) {
        node = node.left;
    }
    return node.data;
}

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

console.log('Searching for 15:', tree.search(15));
console.log('Minimum value:', tree.findMin());

```

```

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         this.root = this._insert(this.root, data);
19     }
20
21     insert(node, data) {
22         if (!node) {
23             return new Node(data);
24         }
25
26         if (data < node.data) {
27             node.left = this._insert(node.left, data);
28         } else if (data > node.data) {
29             node.right = this._insert(node.right, data);
30         }
31
32         return node;
33     }
34 }
```

```
35     // Function to search for a node in the binary tree
36     search(data, node = this.root) {
37         if (!node) {
38             return false;
39         }
40
41         if (data === node.data) {
42             return true;
43         } else if (data < node.data) {
44             return this.search(data, node.left);
45         } else {
46             return this.search(data, node.right);
47         }
48     }
49 }
```

```
50 // Function to find the minimum value in the binary tree
51 findMin(node = this.root) {
52     if (!node) {
53         return null;
54     }
55
56     while (node.left) {
57         node = node.left;
58     }
59     return node.data;
60 }
61 }
62
63 // Example usage
64 const tree = new BinaryTree();
65 tree.insert(10);
66 tree.insert(5);
67 tree.insert(15);
68 tree.insert(3);
69 tree.insert(8);
70
71 console.log('Searching for 15:', tree.search(15));
72 console.log('Minimum value:', tree.findMin());
```

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

```
ls  
node BinaryTree.js
```

The screenshot shows a terminal window with several tabs at the top: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL (which is highlighted with a red border). Below the tabs, the terminal output is displayed in green and blue text. The user runs 'ls' to list files, then executes 'node BinaryTree.js'. The program outputs the search for value 15, the minimum value found (3), and ends with a closing bracket ']'. A small red box highlights the terminal tab.

```
51     findMin(node = this.root) {  
52         if (!node) {  
53             |   return null;  
54         }  
55         while (node.left) {  
  
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    bash +   
  
priyanshurajsim@ip-172-31-35-72:~/Downloads/Simpli$ ls  
BinaryTree.js  
priyanshurajsim@ip-172-31-35-72:~/Downloads/Simpli$ node BinaryTree.js  
Searching for 15: true  
Minimum value: 3  
priyanshurajsim@ip-172-31-35-72:~/Downloads/Simpli$ ]
```

By following these steps, you have successfully implemented and executed binary tree operations in JavaScript. This example uses JavaScript to highlight key techniques such as node searching and identifying the smallest value in a binary tree.