.

Lesson 03 Demo 03

Using Control Flow Statements

Objective: To demonstrate the practical usage of control flow concepts and conditional statements, enhancing their application in decision-making and iterative processes

Tools required: Visual Studio Code and Node.js

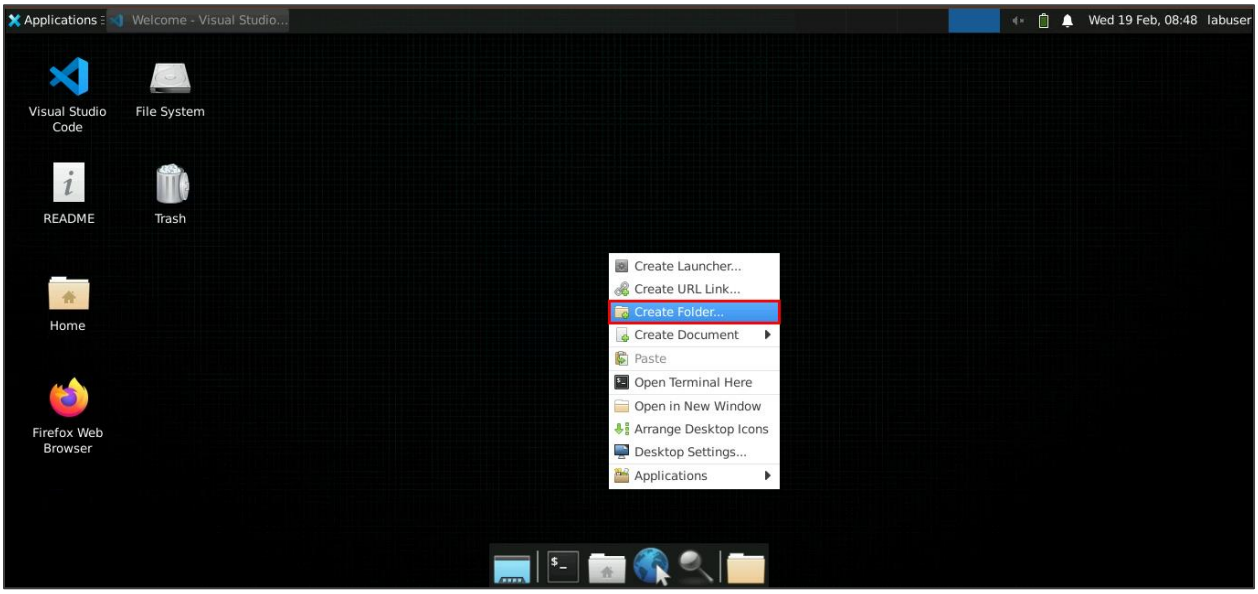
Prerequisites: A basic understanding of control flow statements in JavaScript

Steps to be followed:

1. Create a **ControlFlowStatements.js** folder
2. Execute the JavaScript file

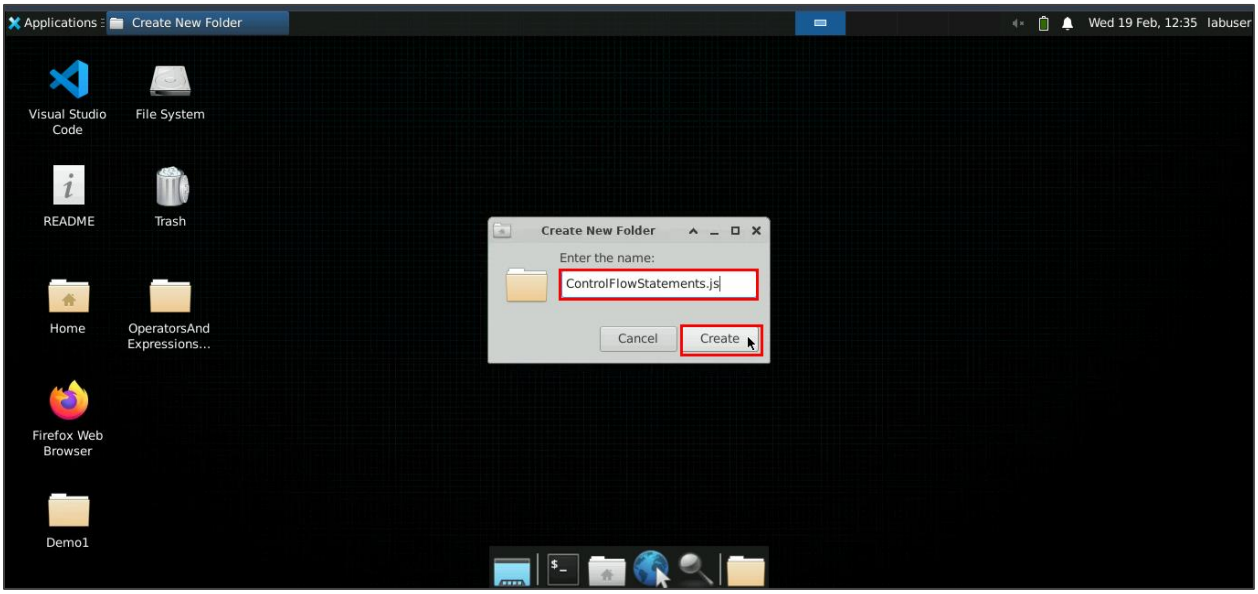
Step 1: Create a ControlFlowStatements.js folder

1.1 Right-click on the desktop and click on **Create Folder…**

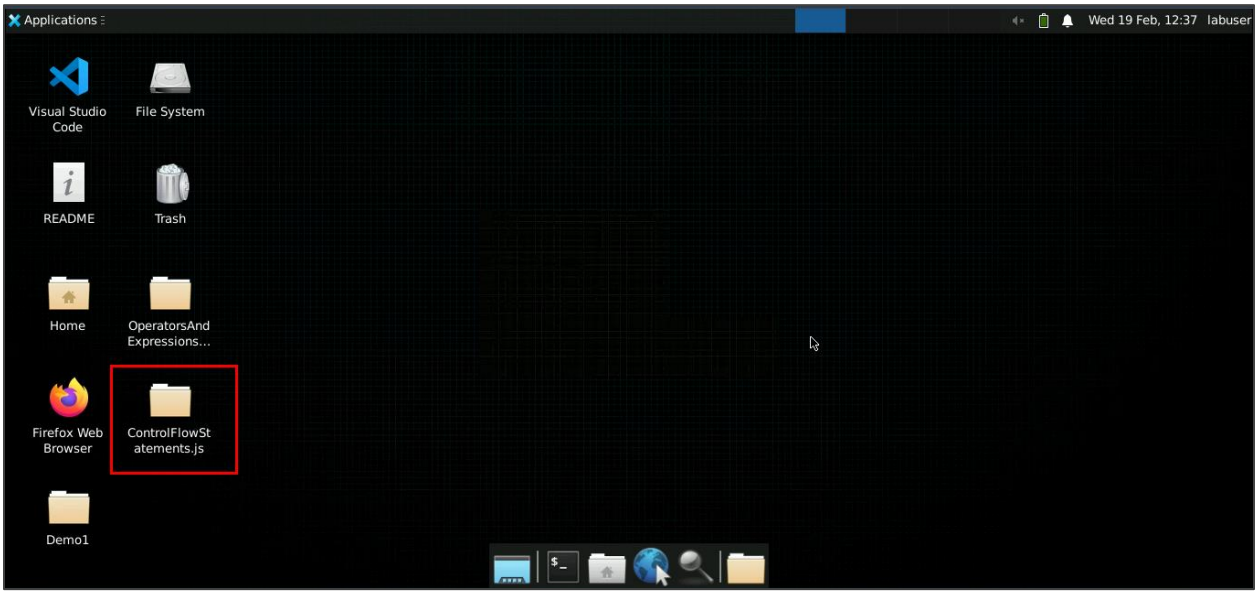


.

1.2 Enter the folder name as **ControlFlowStatements.js** and click on Create



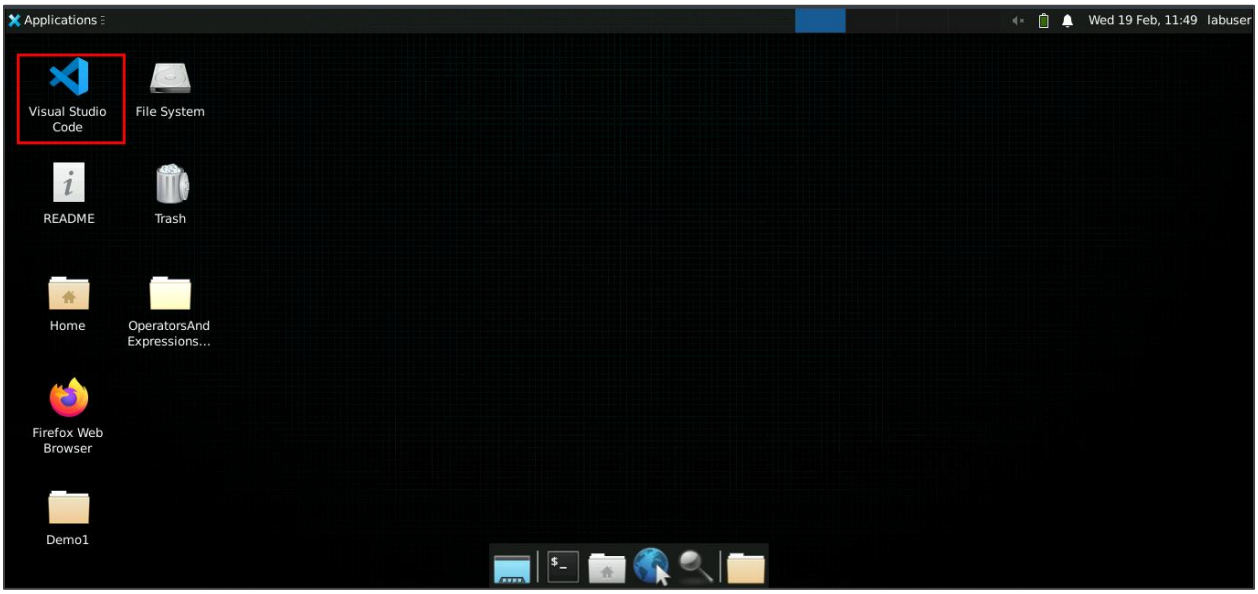
The **ControlFlowStatements.js** folder gets created as shown below:



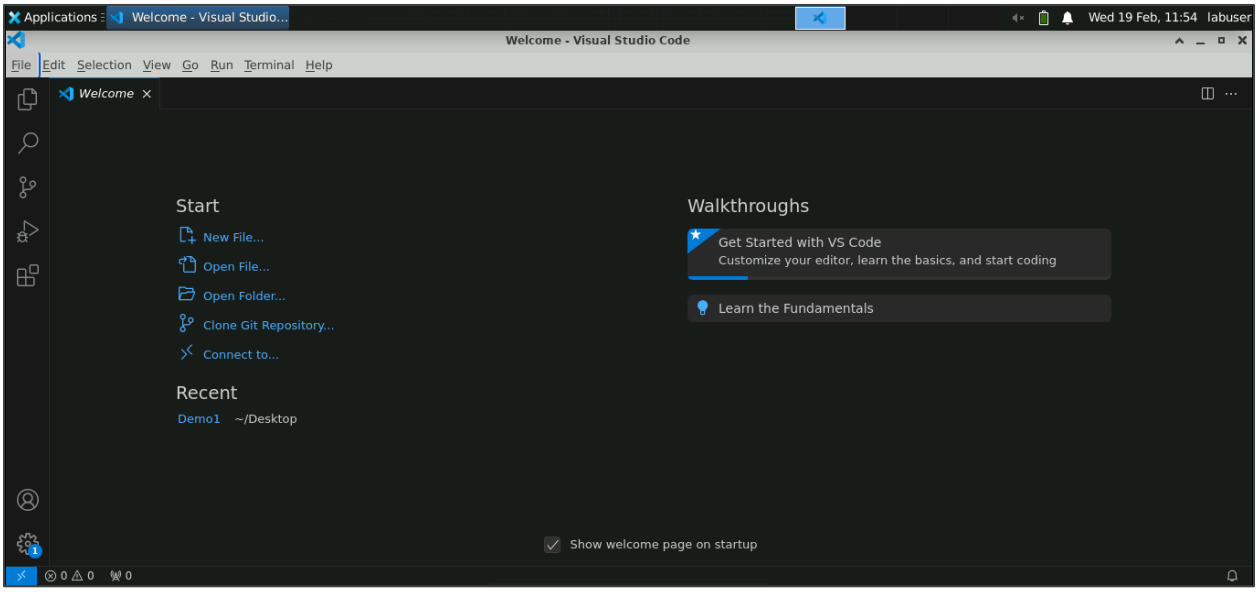
.

Step 2: Execute the JavaScript file

2.1 Double-click on the **Visual Studio Code** icon to open it

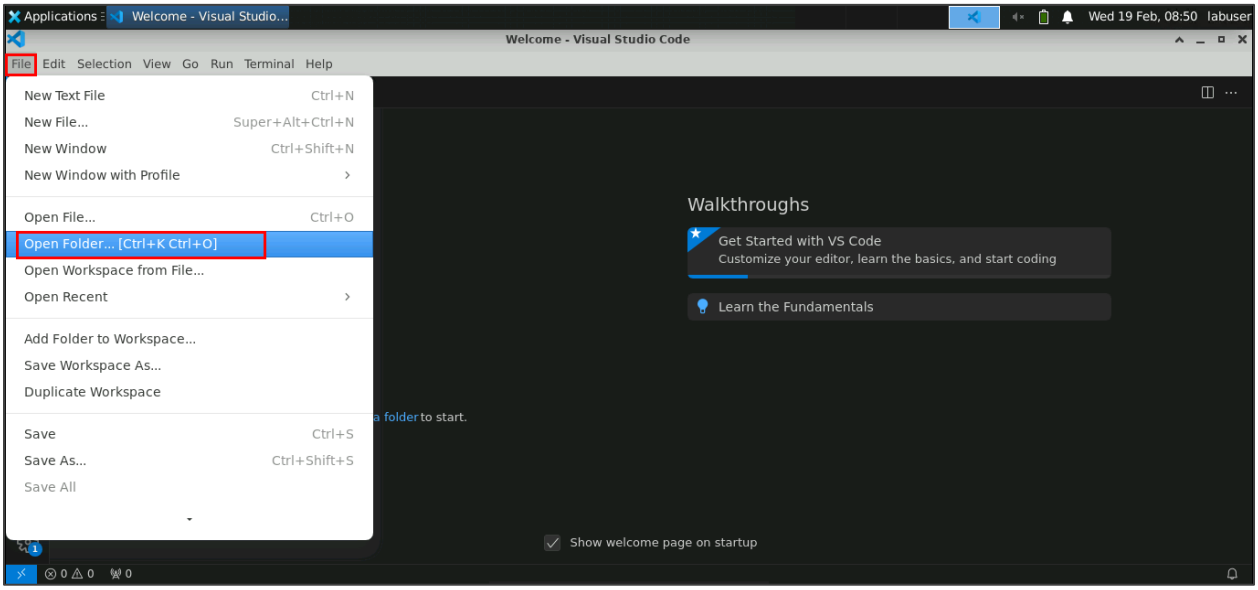


The **Visual Studio Code** opens as shown below:

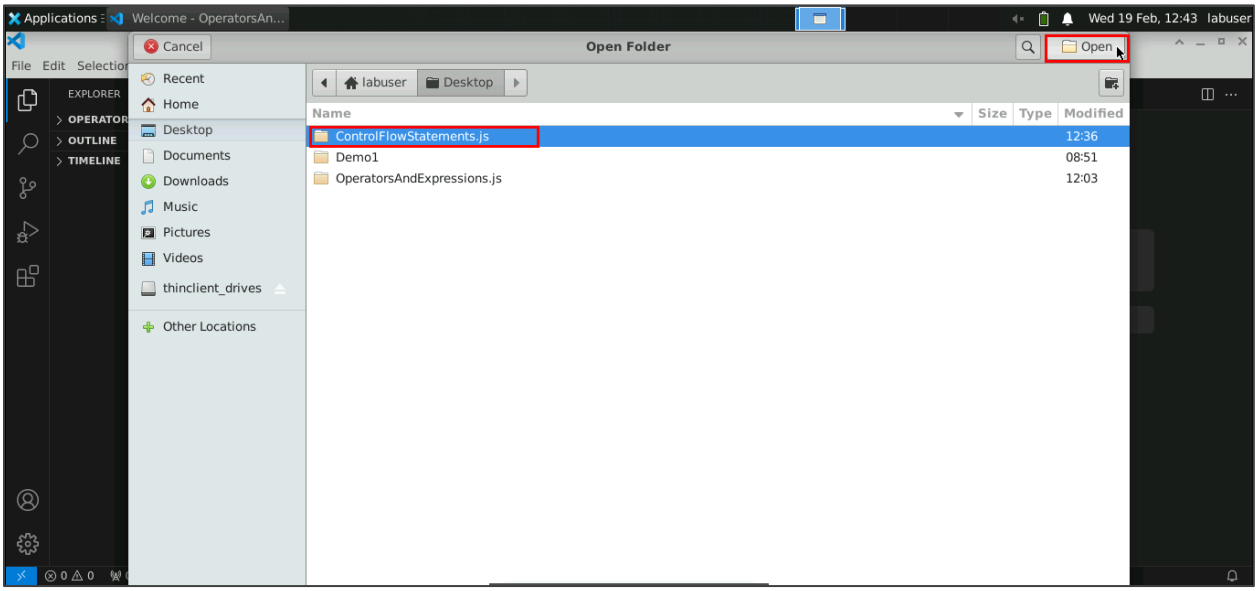


.

2.2 Click on **File**, then click on **Open Folder…**

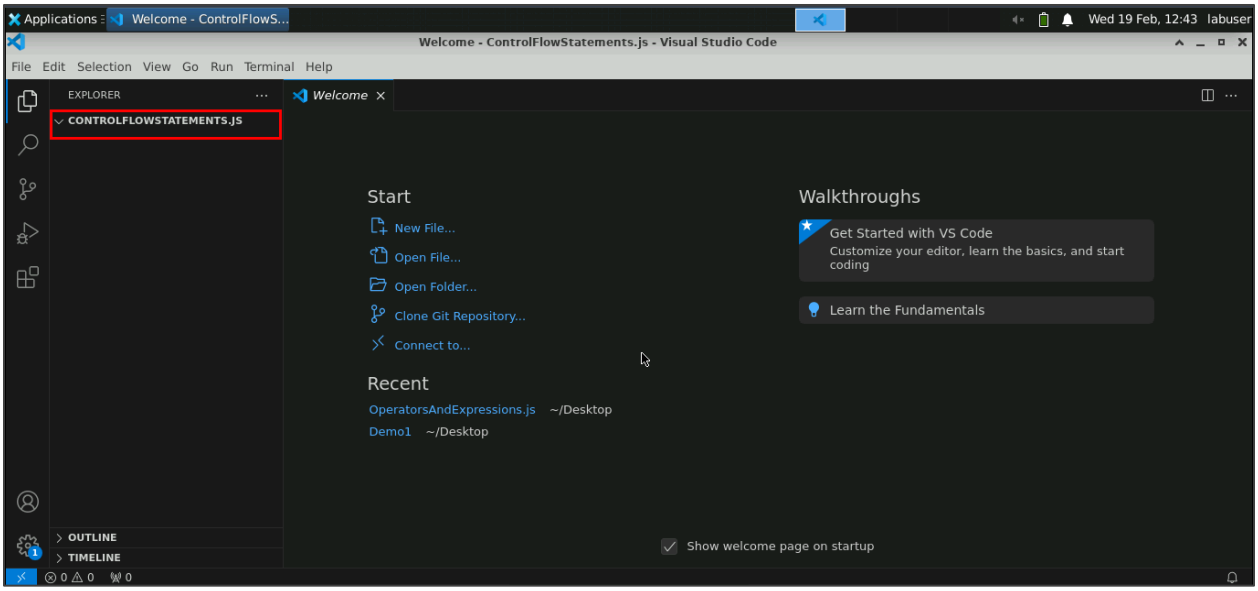


2.3 Select the **ControlFlowStatements.js** folder and click on the **Open** icon to open the folder in Visual Studio Code

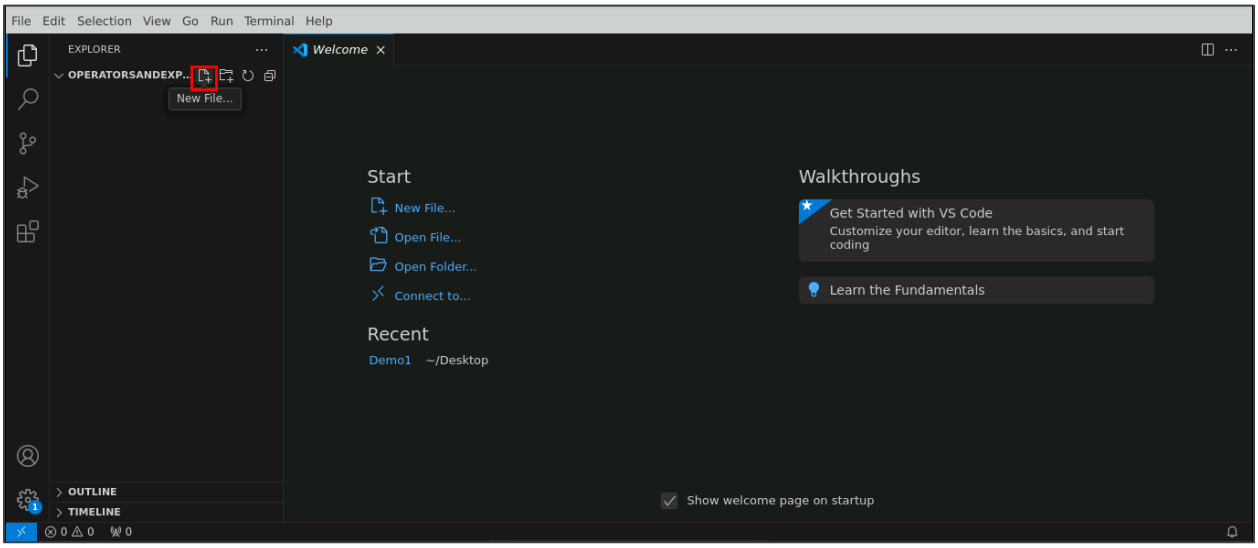


.

The folder opens in **Visual Studio Code** as shown below:

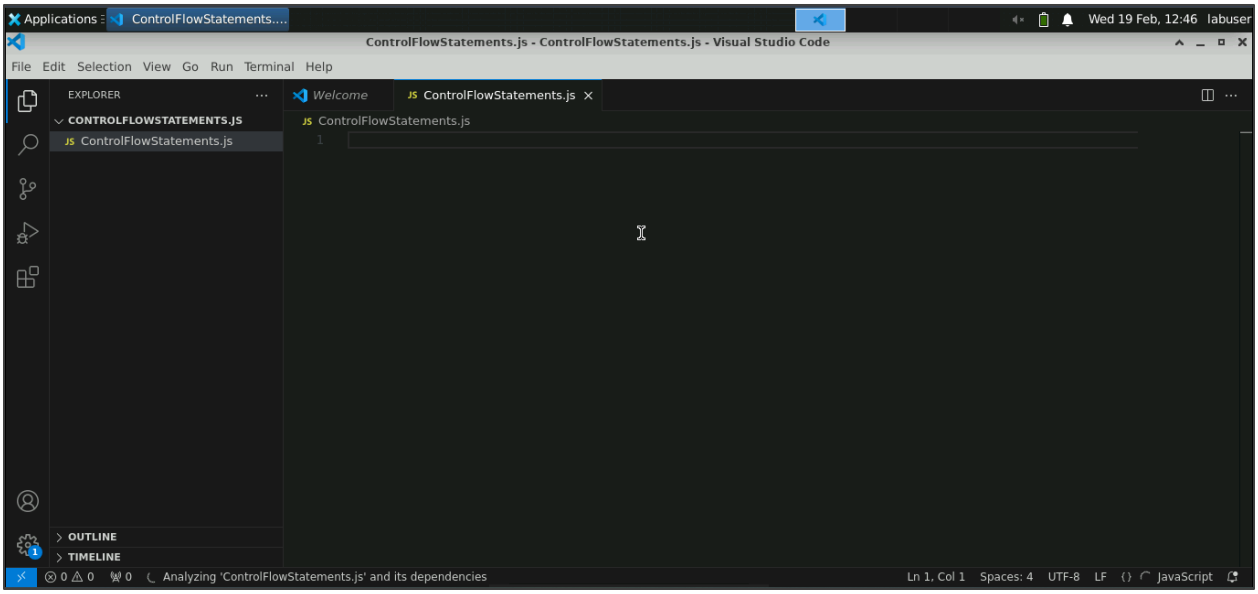


2.4 Click on the **New File** icon to create a new file named **ControlFlowStatements.js**



.

The file gets created as shown below:



2.5 Enter the below code and save the file:

```
let condition = true;

if (condition) {
  console.log("The condition is true.");
} else {
  console.log("The condition is false.");
}

let value = 2;

if (value === 1) {
  console.log("Value is 1.");
} else if (value === 2) {
  console.log("Value is 2.");
} else {
  console.log("Value is neither 1 nor 2.");
}

let fruit = "apple";
```

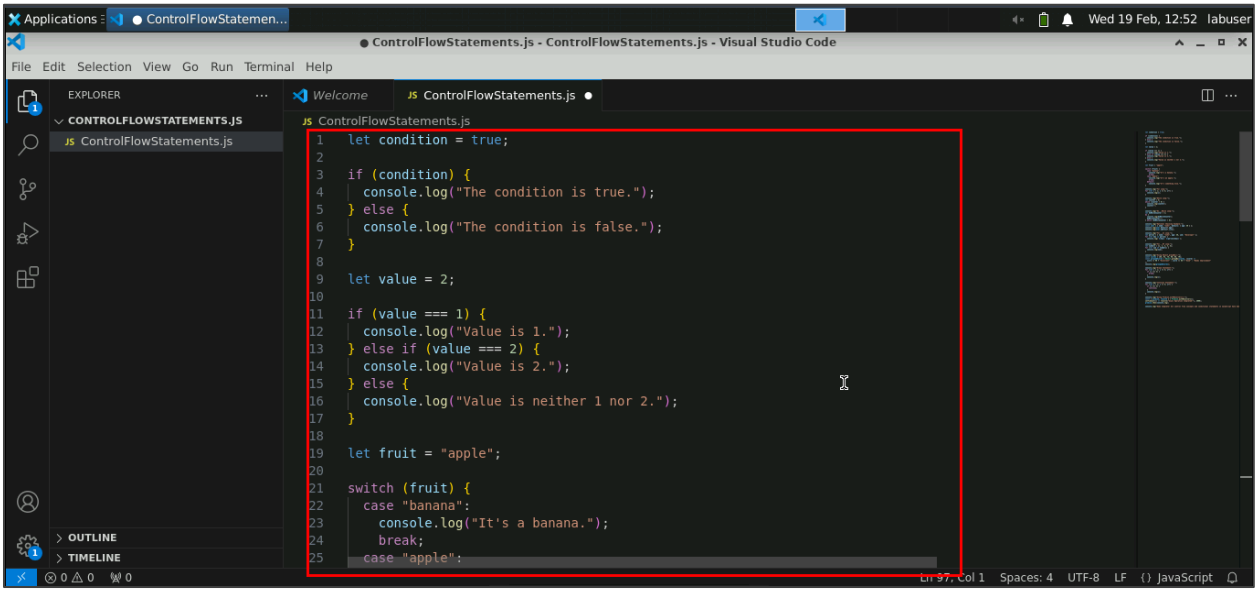
.

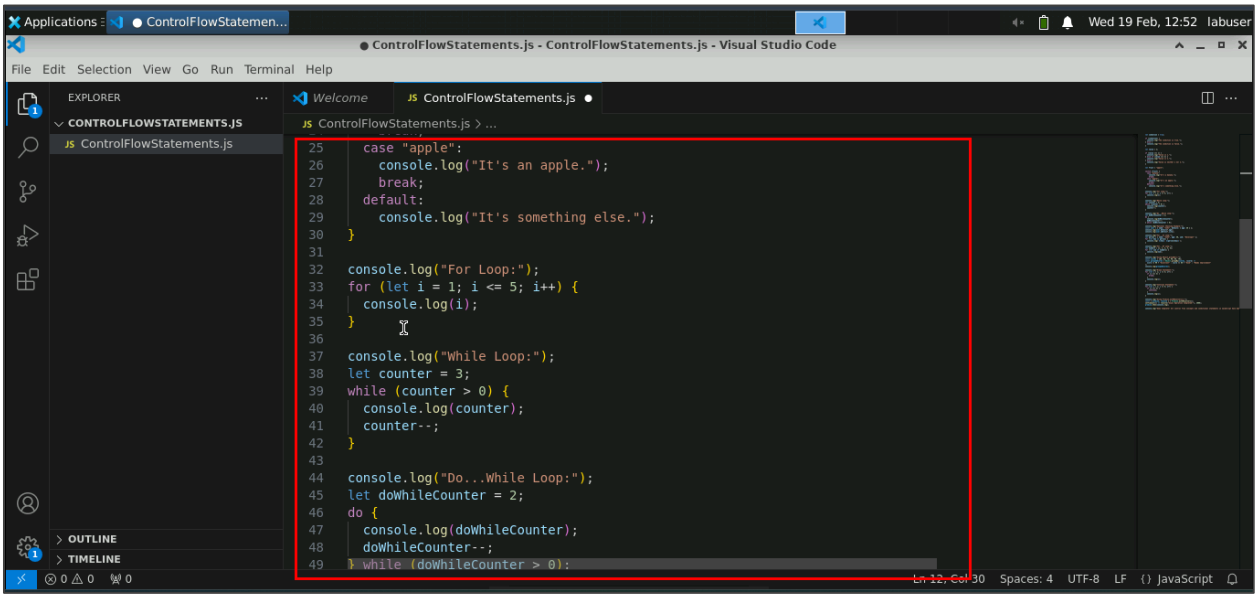
```
switch (fruit) {
  case "banana":
    console.log("It's a banana.");
    break;
  case "apple":
    console.log("It's an apple.");
    break;
  default:
    console.log("It's something else.");
}

console.log("For Loop:");
for (let i = 1; i <= 5; i++) {
  console.log(i);
}

console.log("While Loop:");
let counter = 3;
while (counter > 0) {
  console.log(counter);
  counter--;
}

console.log("Do...While Loop:");
let doWhileCounter = 2;
do {
  console.log(doWhileCounter);
  doWhileCounter--;
} while (doWhileCounter > 0);

console.log("Optional Chaining Example:");
const user = { name: "John", details: { age: 30 } };
console.log(user.details.age);
console.log(user.address.city);

console.log("For...In Loop:");
let person = { name: "John", age: 30, job: "Developer" };
for (let key in person) {
```

```
  console.log(`${key}: ${person[key]}`);
}

console.log("For...Of Loop:");
let numbers = [1, 2, 3, 4, 5];
for (let num of numbers) {
  console.log(num);
}

console.log("Using Object.groupBy():");
const scores = [85, 92, 76, 90, 88, 79];
const groupedScores = Object.groupBy(scores, (score) =>
  score >= 90 ? "Excellent" : score >= 80 ? "Good" : "Needs Improvement"
);
console.log(groupedScores);

console.log("Break Statement:");
for (let i = 1; i <= 5; i++) {
  if (i === 3) {
    break;
  }
  console.log(i);
}

console.log("Continue Statement:");
for (let i = 1; i <= 5; i++) {
  if (i === 3) {
    continue;
  }
  console.log(i);
}

console.log("Using Promise.withResolvers():");
const { promise, resolve } = Promise.withResolvers();
setTimeout(() => resolve("Async Operation Completed!"), 2000);
promise.then(console.log);
```

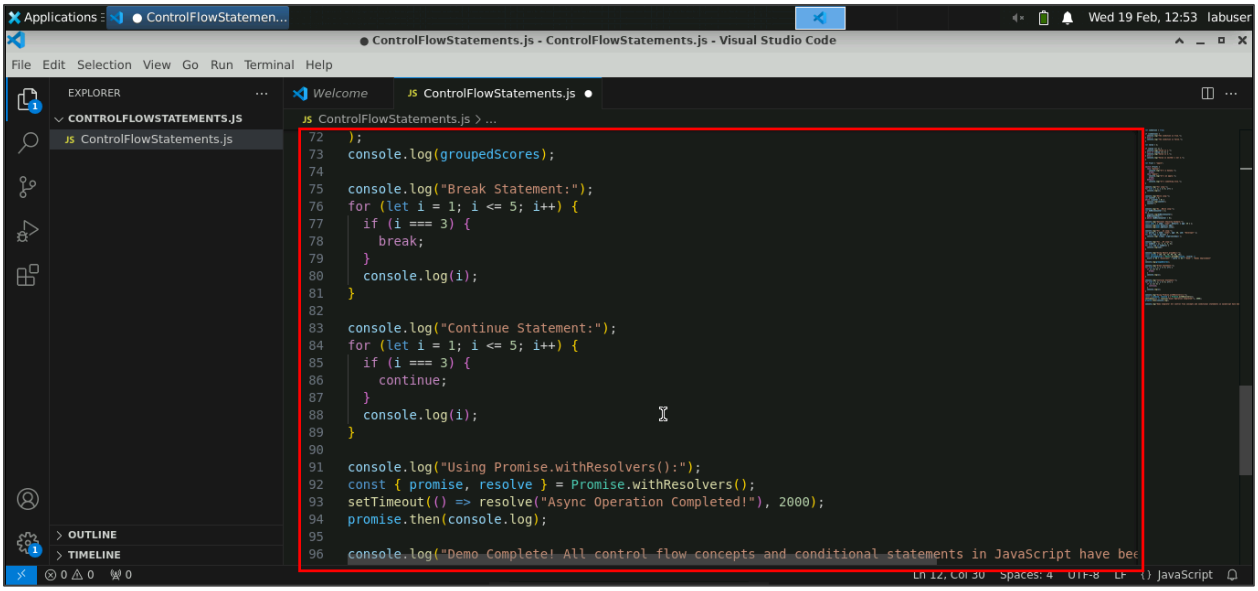
.

console.log("Demo Complete! All control flow concepts and conditional statements in JavaScript have been successfully demonstrated.");



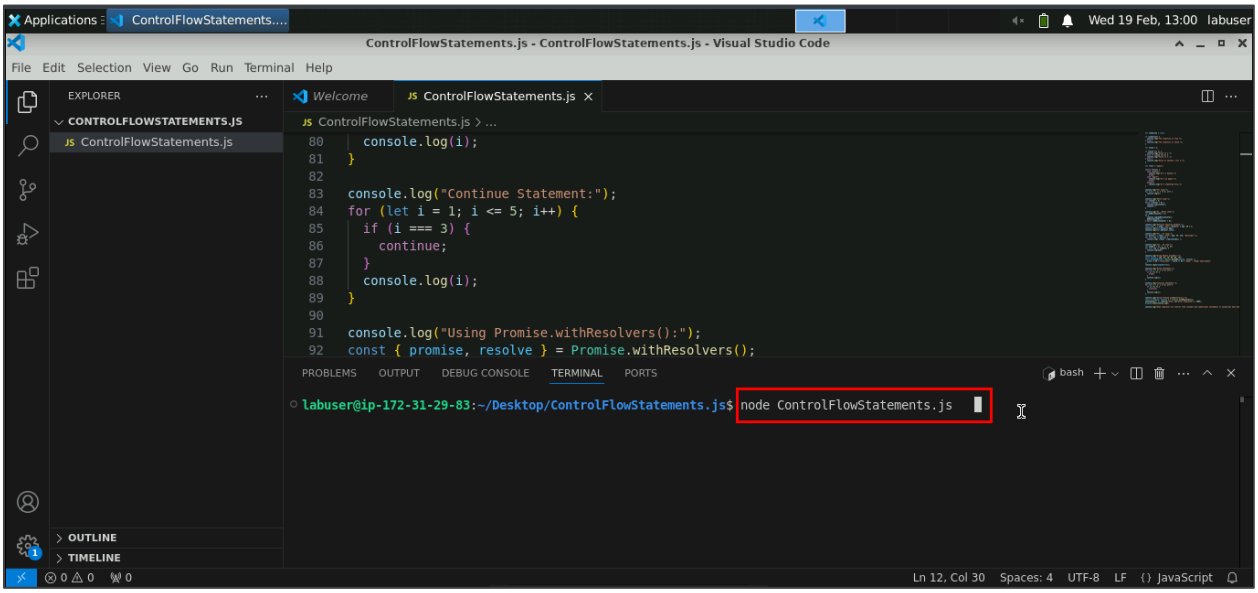


.



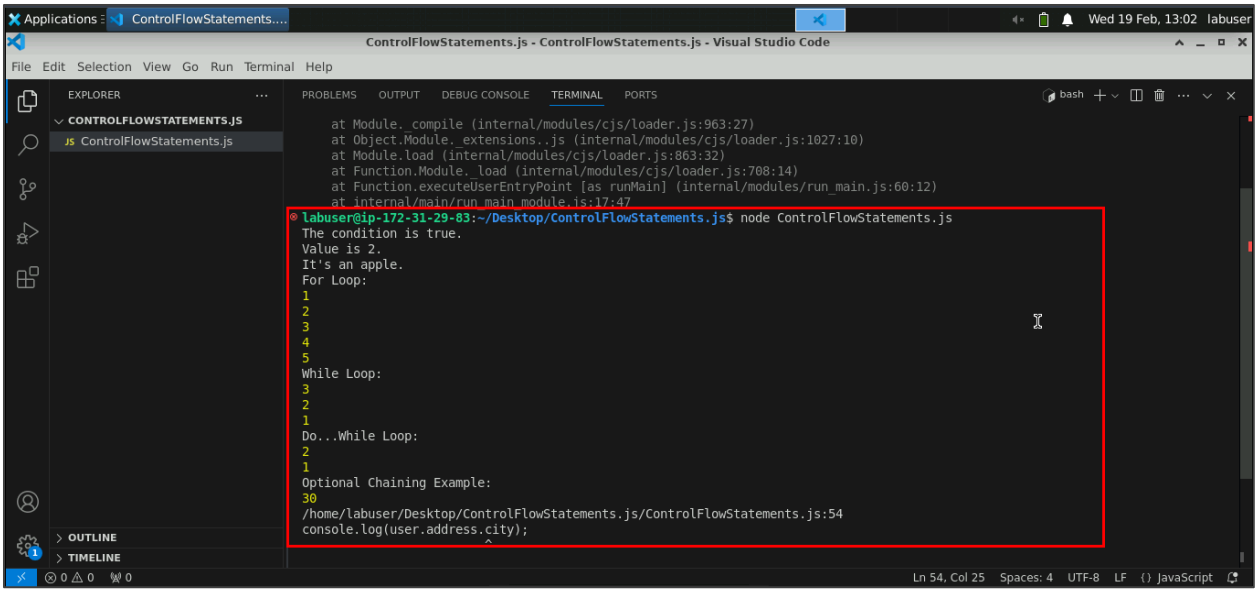
2.6 Open the terminal and run the command below:

node ControlFlowStatements.js



.

The output will appear as shown below:



The above code demonstrates control flow concepts and conditional statements in JavaScript, including if, else if, switch statements, and various looping constructs. It showcases how decision-making structures and loops function to control program execution. Finally, it validates the correctness of these implementations through systematic execution.

By following the above steps, you have successfully demonstrated fundamental control flow concepts in JavaScript, ensuring a clear understanding of decision-making structures and looping mechanisms.