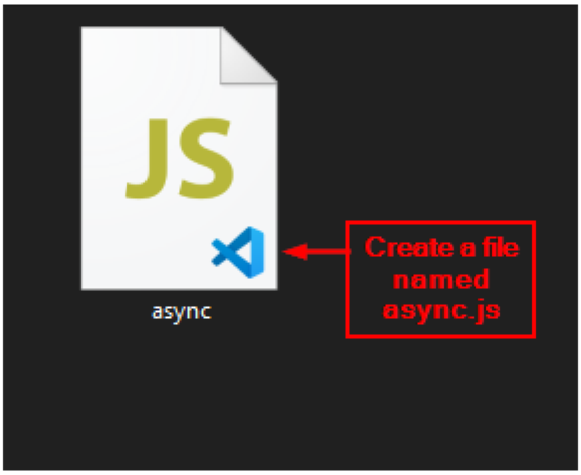
Module 2: Hands-On – 6

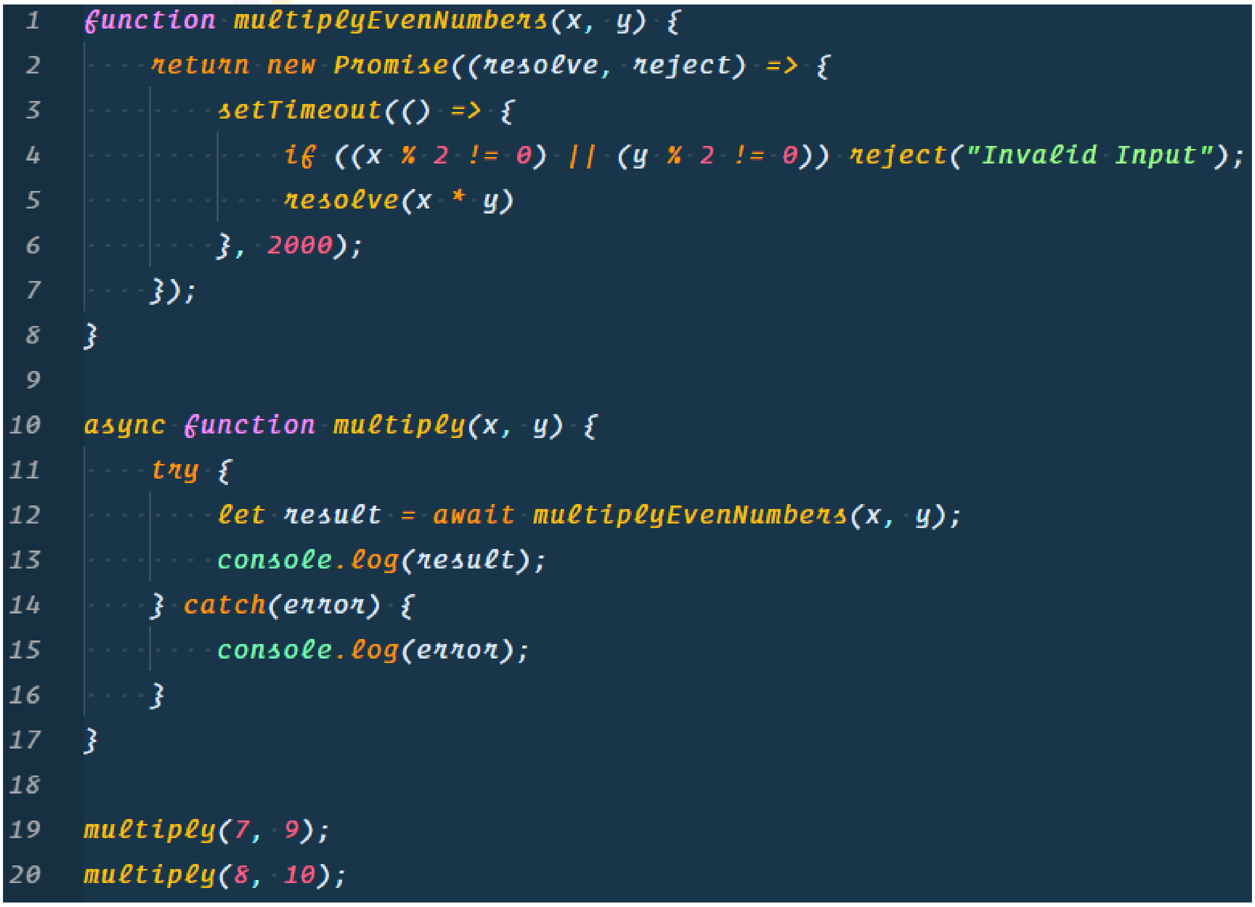
Performing Error Handling Using Async–Await

Step 1: Create a file named async.js



Step 2: Open it in any text editor

Step 3: Type the following code:

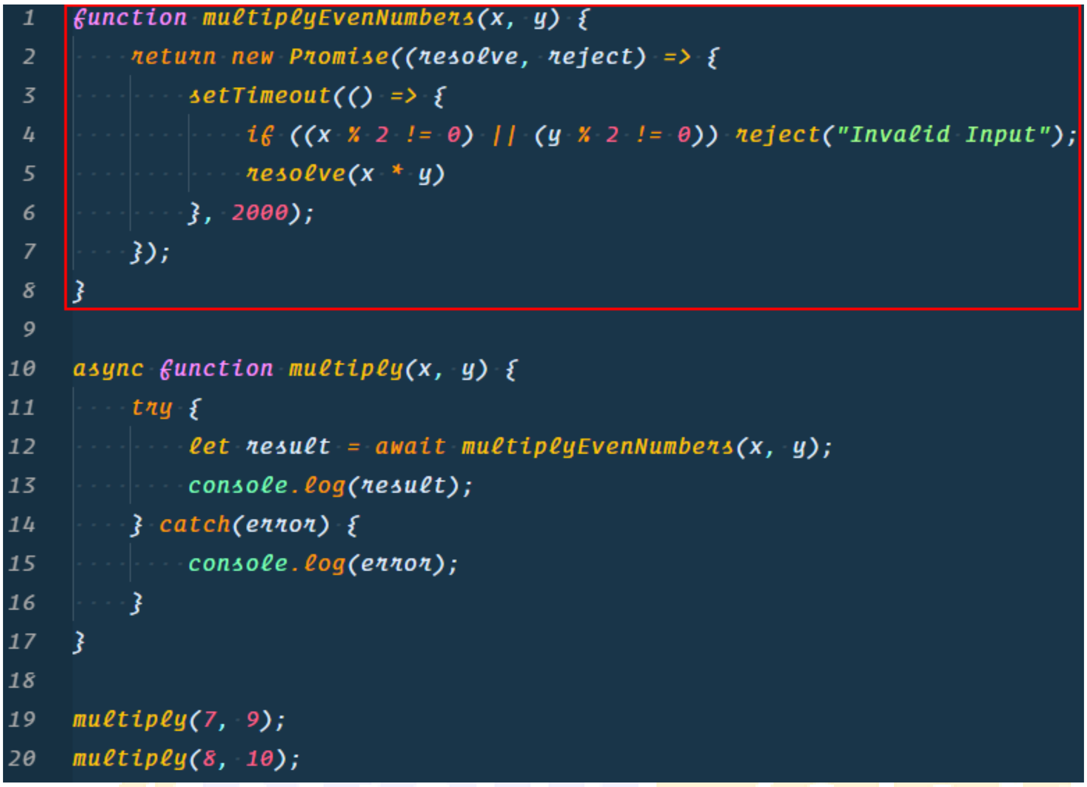


```
function multiplyEvenNumbers(x, y) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            if ((x % 2 != 0) || (y % 2 != 0)) reject("Invalid Input");
            resolve(x * y)
        }, 2000);
    });
}

async function multiply(x, y) {
    try {
        let result = await multiplyEvenNumbers(x, y);
        console.log(result);
    } catch(error) {
        console.log(error);
    }
}

multiply(7, 9);
multiply(8, 10);
```

Step 3.1: Create a function named multiplyEvenNumbers that takes two numbers as arguments: ***x*** and ***y*** return a new promise that waits for 2 seconds and then checks if either x or y is odd. If they are, then it calls the reject function with the error 'Invalid Input'; else, it will call the resolve function with the result as x * y

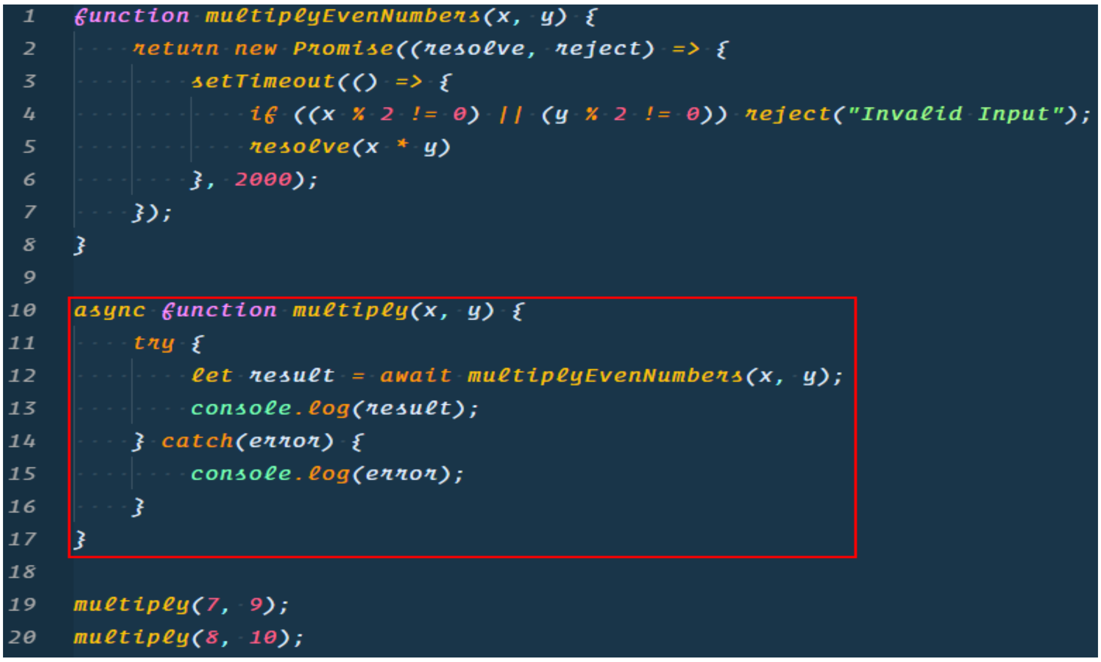


```
function multiplyEvenNumbers(x, y) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            if ((x % 2 != 0) || (y % 2 != 0)) reject("Invalid Input");
            resolve(x * y)
        }, 2000);
    });
}

async function multiply(x, y) {
    try {
        let result = await multiplyEvenNumbers(x, y);
        console.log(result);
    } catch(error) {
        console.log(error);
    }
}

multiply(7, 9);
multiply(8, 10);
```

Step 3.2: Create another function named **multiple** with the async keyword. The method uses the try–catch block to handle errors. In the try block, it calls the multiplyEvenNumbers function and waits for the promise to resolve and return the answer which it logs to console. In the catch block, if any error occurs it logs the error

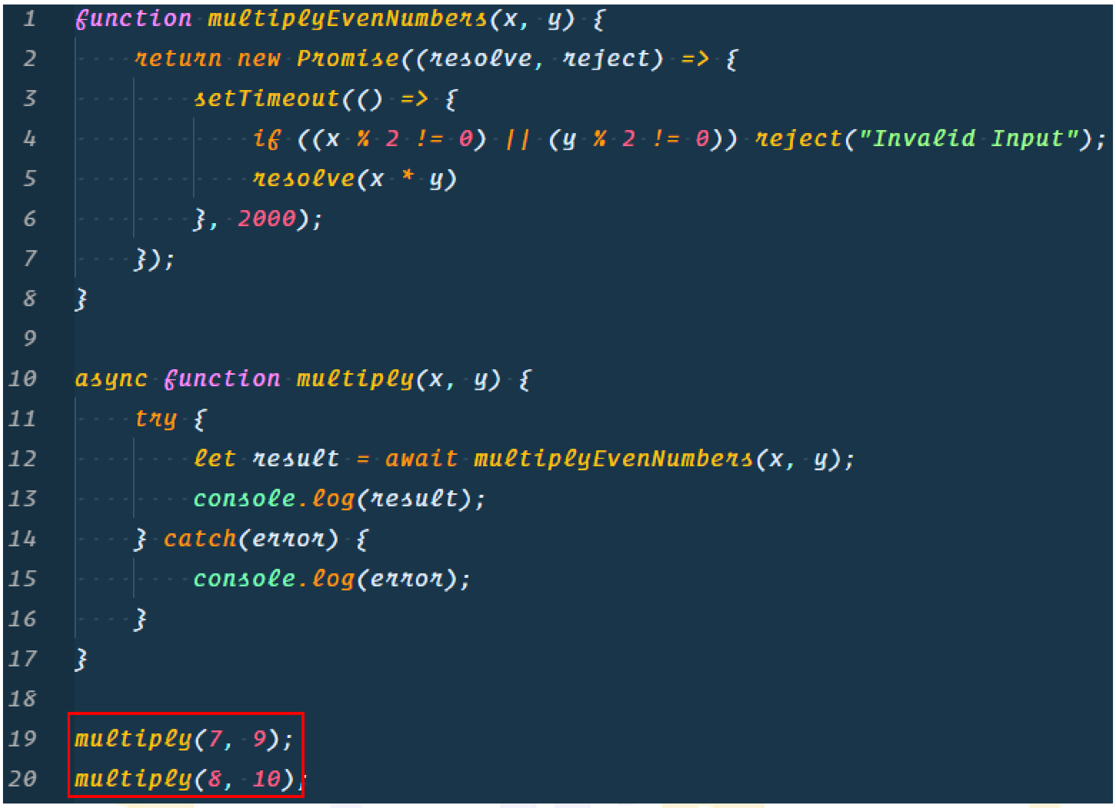


```
function multiplyEvenNumbers(x, y) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            if ((x % 2 != 0) || (y % 2 != 0)) reject("Invalid Input");
            resolve(x * y)
        }, 2000);
    });
}

async function multiply(x, y) {
    try {
        let result = await multiplyEvenNumbers(x, y);
        console.log(result);
    } catch(error) {
        console.log(error);
    }
}

multiply(7, 9);
multiply(8, 10);
```

Step 3.3: Call the multiplyEvenNumbers function two times once with an invalid input (odd numbers) and again with a valid input (even numbers)

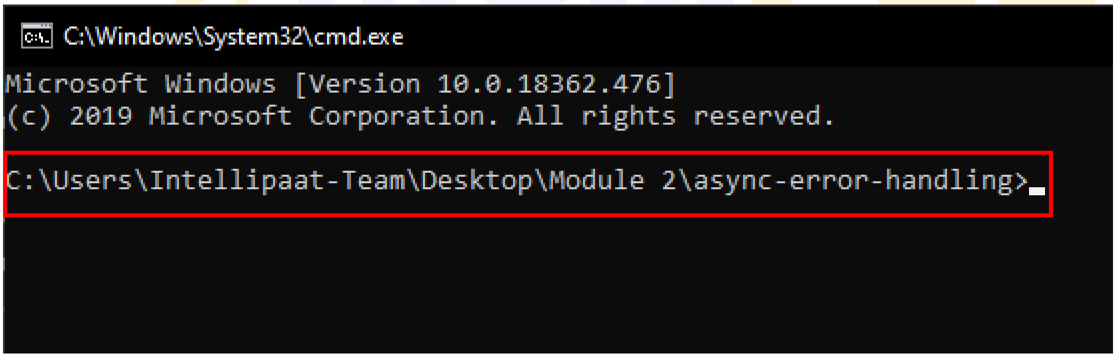


```
function multiplyEvenNumbers(x, y) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            if ((x % 2 != 0) || (y % 2 != 0)) reject("Invalid Input");
            resolve(x * y)
        }, 2000);
    });
}

async function multiply(x, y) {
    try {
        let result = await multiplyEvenNumbers(x, y);
        console.log(result);
    } catch(error) {
        console.log(error);
    }
}

multiply(7, 9);
multiply(8, 10);
```

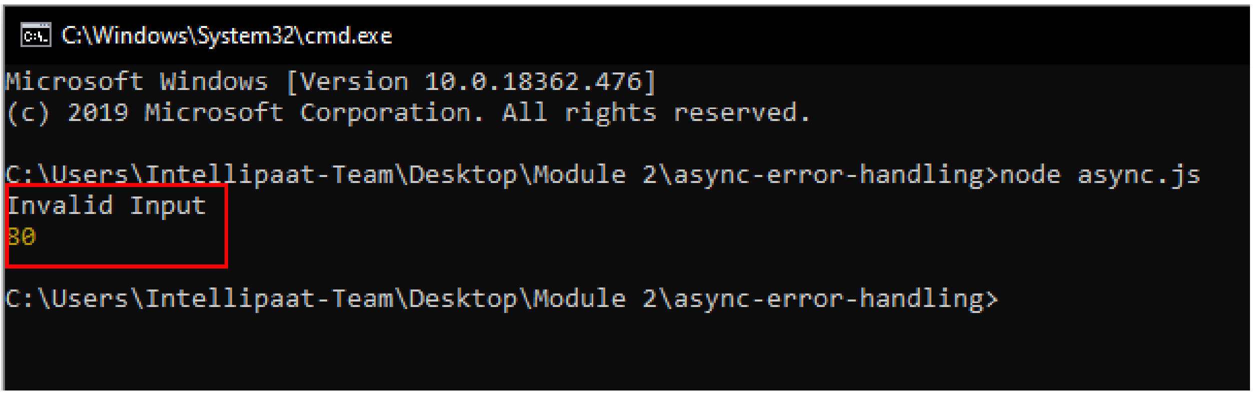
Step 4: Open the command prompt in the same directory as the file



```
Microsoft Windows [Version 10.0.18362.476]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Intellipaat-Team\Desktop\Module 2\async-error-handling>_
```

Step 5: Run the file using the command 'node async.js'



```
Microsoft Windows [Version 10.0.18362.476]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Intellipaat-Team\Desktop\Module 2\async-error-handling>node async.js
Invalid Input
80

C:\Users\Intellipaat-Team\Desktop\Module 2\async-error-handling>
```