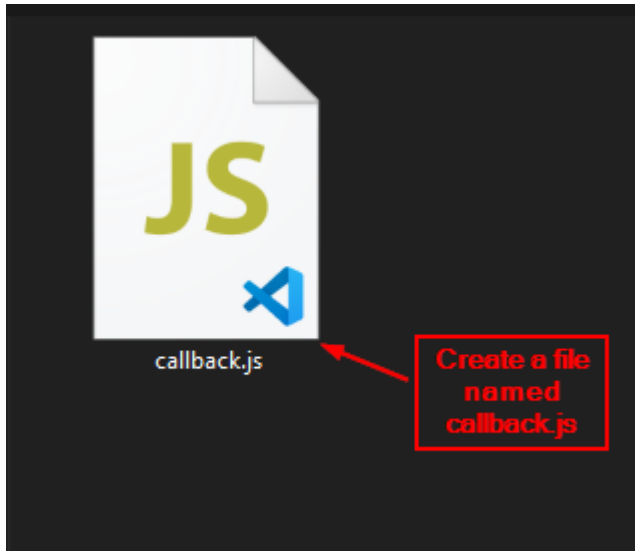# Module 2: Hands-On – 4
Performing Error Handling Using Callbacks

Step 1: Create a file named callback.js

Step 2: Open it in any text editor

Step 3: Type the following code:

```js
function multiplyEvenNumbers(x, y, callback) {
    if ((x % 2 != 0) || (y % 2 != 0)) setTimeout(() => callback("Invalid Input"), 2000);
    else setTimeout(() => callback(null, x * y), 2000);
}

multiplyEvenNumbers(7, 9, (error, result) => {
    if (error != null) {
        console.log(error);
        return;
    }
    console.log(result);
});

multiplyEvenNumbers(8, 10, (error, result) => {
    if (error != null) {
        console.log(error);
        return;
    }
    console.log(result);
});
```

Step 3.1: Create a function named **multiplyEvenNumbers** that takes three arguments: two numbers *x* and *y* and a callback function. If either x or y is odd, then it waits for 2 seconds and calls the callback function with the error 'Invalid Input'; else, it will waits for 2 seconds and call the callback function with the error as null and the result as x * y

```javascript
function multiplyEvenNumbers(x, y, callback) {
    if ((x % 2 != 0) || (y % 2 != 0)) setTimeout(() => callback("Invalid Input"), 2000);
    else setTimeout(() => callback(null, x * y), 2000);
}

multiplyEvenNumbers(7, 9, (error, result) => {
    if (error != null) {
        console.log(error);
        return;
    }
    console.log(result);
});

multiplyEvenNumbers(8, 10, (error, result) => {
    if (error != null) {
        console.log(error);
        return;
    }
    console.log(result);
});
```
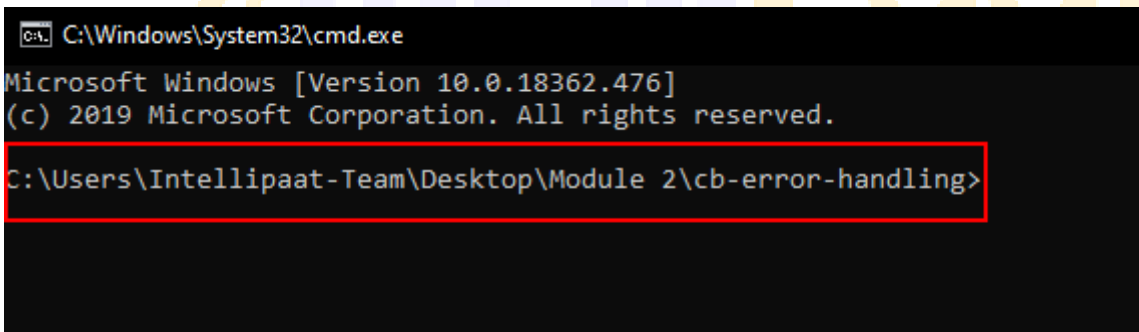
Step 3.2: Call the multiplyEvenNumbers function with an invalid input (odd numbers) and a callback that logs the error if there is an error, or logs the result if there is no error

```javascript
function multiplyEvenNumbers(x, y, callback) {
    if ((x % 2 != 0) || (y % 2 != 0)) setTimeout(() => callback("Invalid Input"), 2000);
    else setTimeout(() => callback(null, x * y), 2000);
}

multiplyEvenNumbers(7, 9, (error, result) => {
    if (error != null) {
        console.log(error);
        return;
    }
    console.log(result);
});

multiplyEvenNumbers(8, 10, (error, result) => {
    if (error != null) {
        console.log(error);
        return;
    }
    console.log(result);
});
```

Step 3.3: Call the multiplyEvenNumbers function again with a valid input (even numbers) and a callback that logs the error if there is an error, or logs the result if there is no error

```
1   function multiplyEvenNumbers(x, y, callback) {
2       if ((x % 2 != 0) || (y % 2 != 0)) setTimeout(() => callback("Invalid Input"), 2000);
3       else setTimeout(() => callback(null, x * y), 2000);
4   }
5
6   multiplyEvenNumbers(7, 9, (error, result) => {
7       if (error != null) {
8           console.log(error);
9           return;
10      }
11      console.log(result);
12  });
13
14  multiplyEvenNumbers(8, 10, (error, result) => {
15      if (error != null) {
16          console.log(error);
17          return;
18      }
19      console.log(result);
20  });
```

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

```
C:\Windows\System32\cmd.exe

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

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

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

```
C:\Windows\System32\cmd.exe

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

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