# Lesson 06 Demo 01

# Executing File System Commands

**Objective:** To execute file system operations like reading, writing, and deleting to manage files within a file system

**Tools required:** Linux Cent OS, Node Package Manager, and Visual Studio Code

**Prerequisites:** Basic Linux Commands, NPM commands, JavaScript, and file system module

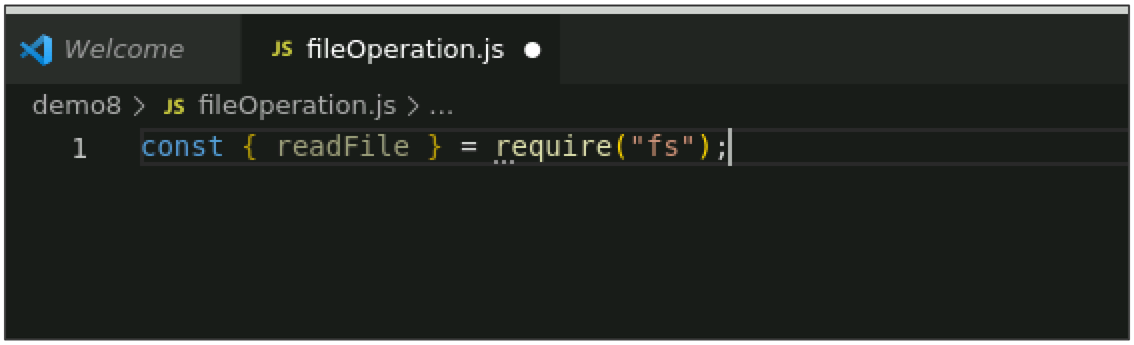
Steps to be followed:

1. Read files using the **fs.readFile()** method
2. Write into the files using the **fs.writeFile()** method
3. Delete files using the **unlink(path, callback)** method

## Step 1: Read files using the fs.readFile() method

1.1 Import the file system module in the **fileOperation.js** file:

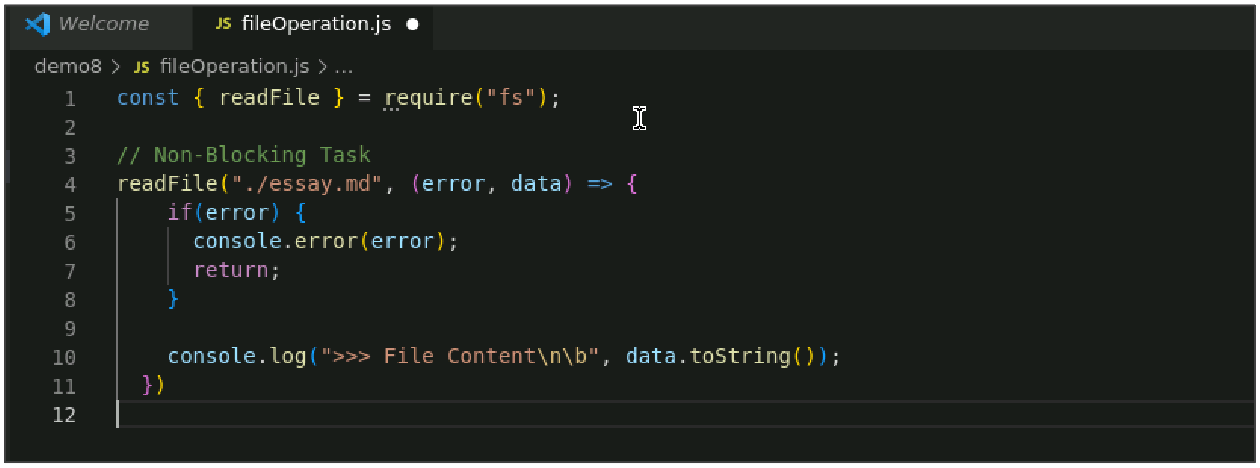
**const { readFile } = require("fs");**



```
const { readFile } = require("fs");
```

1.2 Write the following non-blocking task that reads the **essay.md** file from the system and print the contents on the console:

```
// Non-Blocking Task
readFile("./essay.md", (error, data) => {
 if(error) {
  console.error(error);
  return;
 }

 console.log(">>> File Content\n\b", data.toString());
})
```



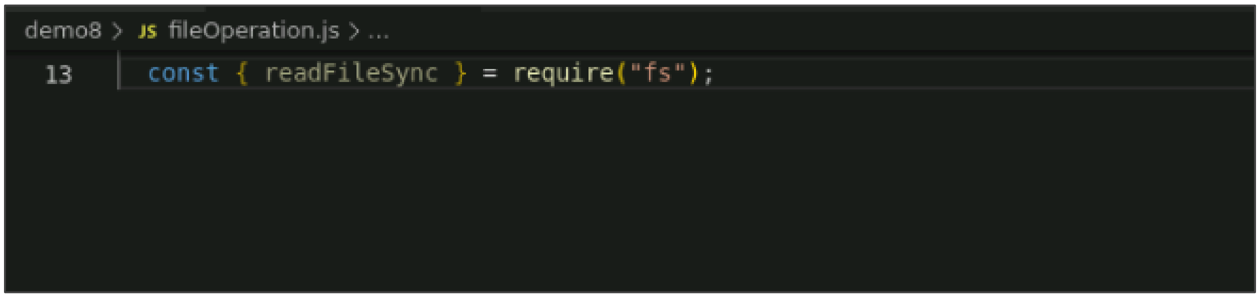
Welcome | fileOperation.js

demo8 > fileOperation.js > ...

```
const { readFile } = require("fs");

// Non-Blocking Task
readFile("./essay.md", (error, data) => {
    if(error) {
      console.error(error);
      return;
    }

    console.log(">>> File Content\n\b", data.toString());
  })

```

1.3 Use the following code statement to read the file synchronously:

```
const { readFileSync } = require("fs");
```

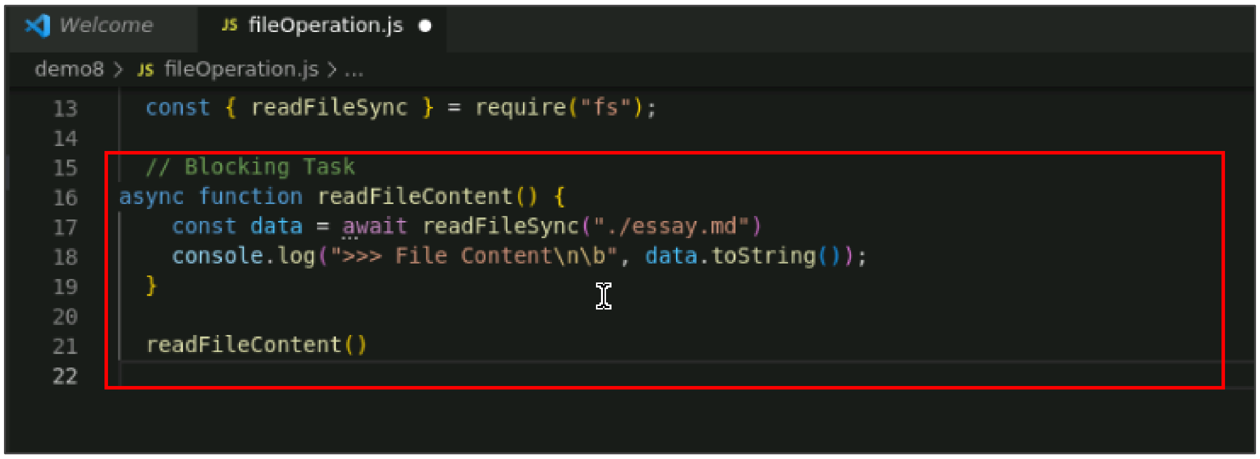


demo8 > fileOperation.js > ...

```
const { readFileSync } = require("fs");
```

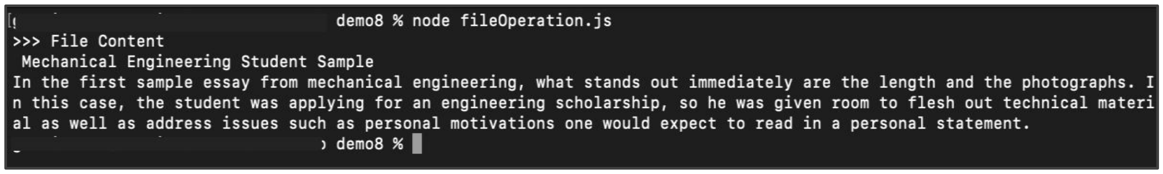
1.4 Write the blocking task that reads the **essay.md** file from the system using the **readFileSync** function and print the contents on the console

```
// Blocking Task
async function readFileContent() {
  const data = await readFileSync("./essay.md")
  console.log(">>> File Content\n\b", data.toString());
}

readFileContent()
```



```
const { readFileSync } = require("fs");

// Blocking Task
async function readFileContent() {
    const data = await readFileSync("./essay.md")
    console.log(">>> File Content\n\b", data.toString());
}

readFileContent()
```

1.5 Execute the following code in the terminal to view the output:
**node fileOperations.js**

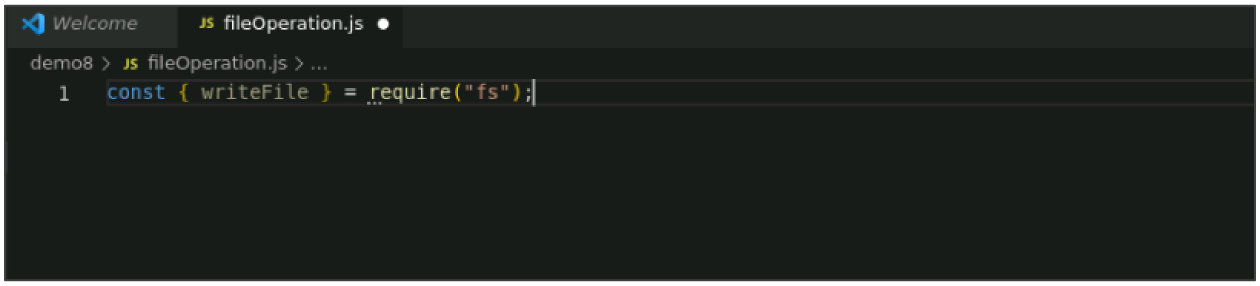


```
[redacted] demo8 % node fileOperation.js
>>> File Content
 Mechanical Engineering Student Sample
In the first sample essay from mechanical engineering, what stands out immediately are the length and the photographs. I
n this case, the student was applying for an engineering scholarship, so he was given room to flesh out technical materi
al as well as address issues such as personal motivations one would expect to read in a personal statement.
[redacted] demo8 % 
```

## Step 2: Write into the files using the fs.writeFile() method

2.1 Import the file system module in the **fileOperation.js** file:

```
const { writeFile } = require("fs");
```



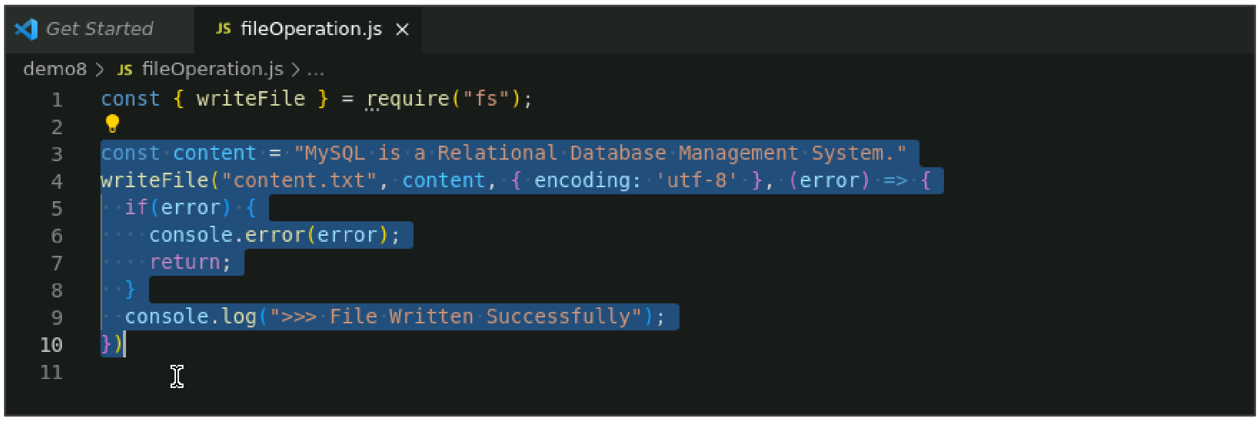
Welcome | fileOperation.js

demo8 > fileOperation.js > ...

```
const { writeFile } = require("fs");
```

2.2 Add the following code snippet to write a string into a file in async mode:

```
const content = "MySQL is a Relational Database Management System."
writeFile("content.txt", content, { encoding: 'utf-8' }, (error) => {
  if(error) {
    console.error(error);
    return;
  }
  console.log(">>> File Written Successfully");
})
```



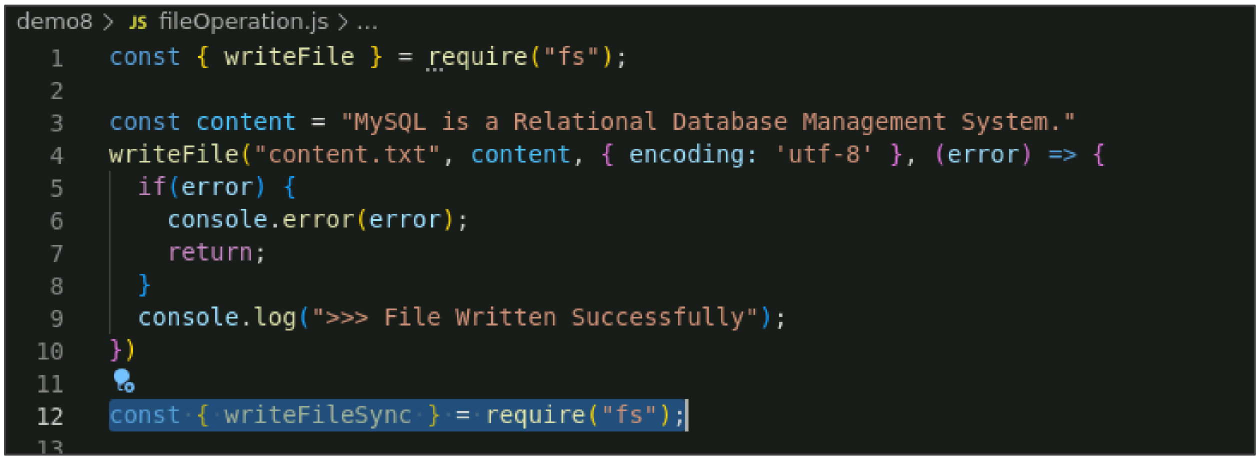
Get Started | fileOperation.js

demo8 > fileOperation.js > ...

```
const { writeFile } = require("fs");

const content = "MySQL is a Relational Database Management System."
writeFile("content.txt", content, { encoding: 'utf-8' }, (error) => {
  if(error) {
    console.error(error);
    return;
  }
  console.log(">>> File Written Successfully");
})
```

2.3 Import the file system module again to write into the a file in synchronous mode:

**const { writeFileSync } = require("fs");**

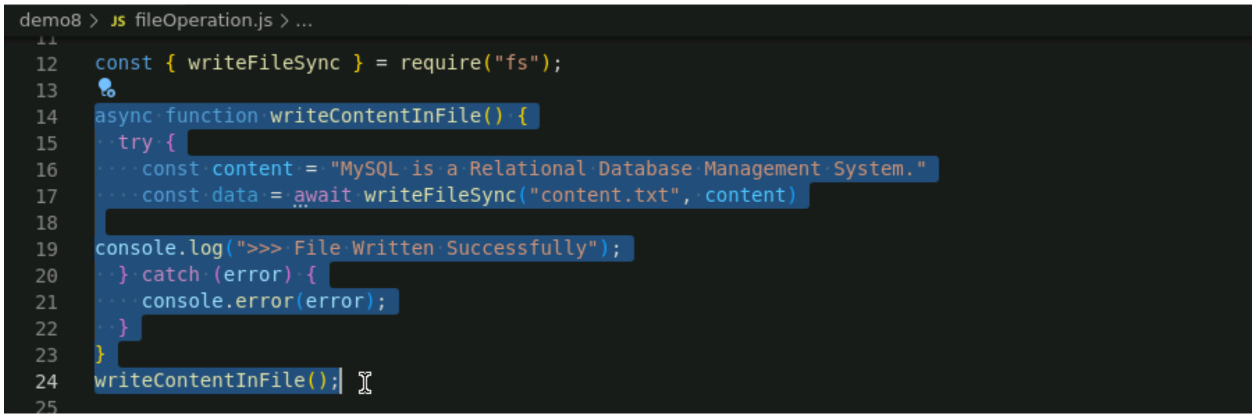


demo8 > JS fileOperation.js > ...

```
const { writeFile } = require("fs");

const content = "MySQL is a Relational Database Management System."
writeFile("content.txt", content, { encoding: 'utf-8' }, (error) => {
  if(error) {
    console.error(error);
    return;
  }
  console.log(">>> File Written Successfully");
})

const { writeFileSync } = require("fs");
```

2.4 Add the following code snippet to write a string into a file in synchronous mode:

```
async function writeContentInFile() {
  try {
    const content = "MySQL is a Relational Database Management System."
    const data = await writeFileSync("content.txt", content)

console.log(">>> File Written Successfully");
  } catch (error) {
    console.error(error);
  }
}
writeContentInFile();
```



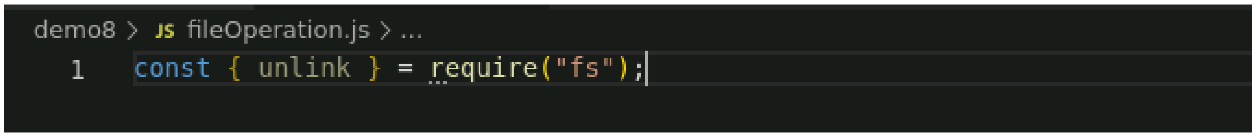
demo8 > JS fileOperation.js > ...

```
const { writeFileSync } = require("fs");

async function writeContentInFile() {
  try {
    const content = "MySQL is a Relational Database Management System."
    const data = await writeFileSync("content.txt", content)

console.log(">>> File Written Successfully");
  } catch (error) {
    console.error(error);
  }
}
writeContentInFile();
```

## Step 3: Delete files using the unlink(path, callback) method

3.1 Import the file system module:

**const { unlink } = require("fs");**

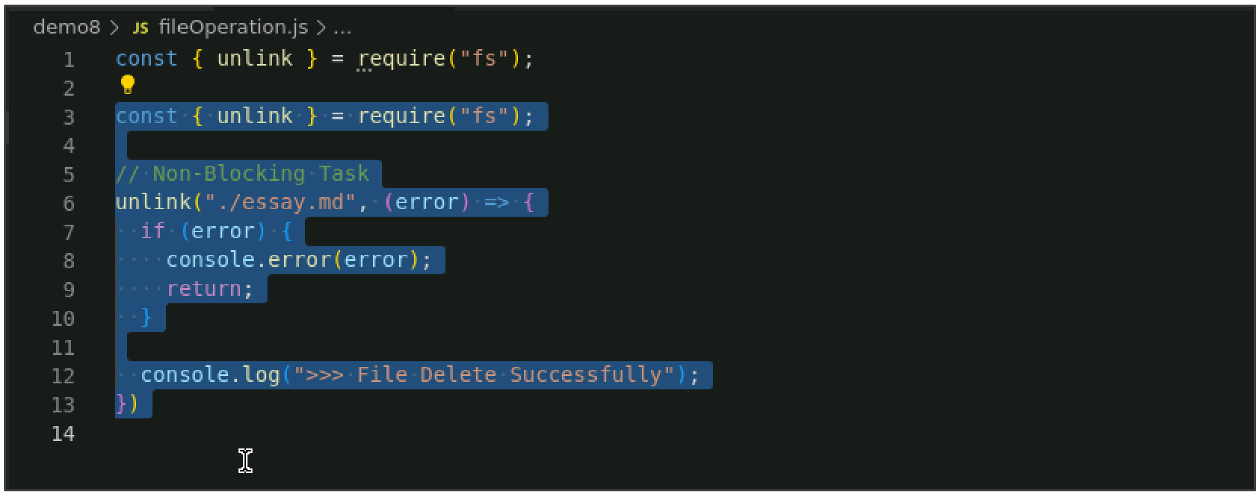


demo8 > JS fileOperation.js > ...

```
const { unlink } = require("fs");
```

3.2 Use the **unlink** function and pass the file as an input parameter for deletion in async mode:

```
const { unlink } = require("fs");

// Non-Blocking Task
unlink("./essay.md", (error) => {
 if (error) {
   console.error(error);
   return;
 }

 console.log(">>> File Delete Successfully");
})
```

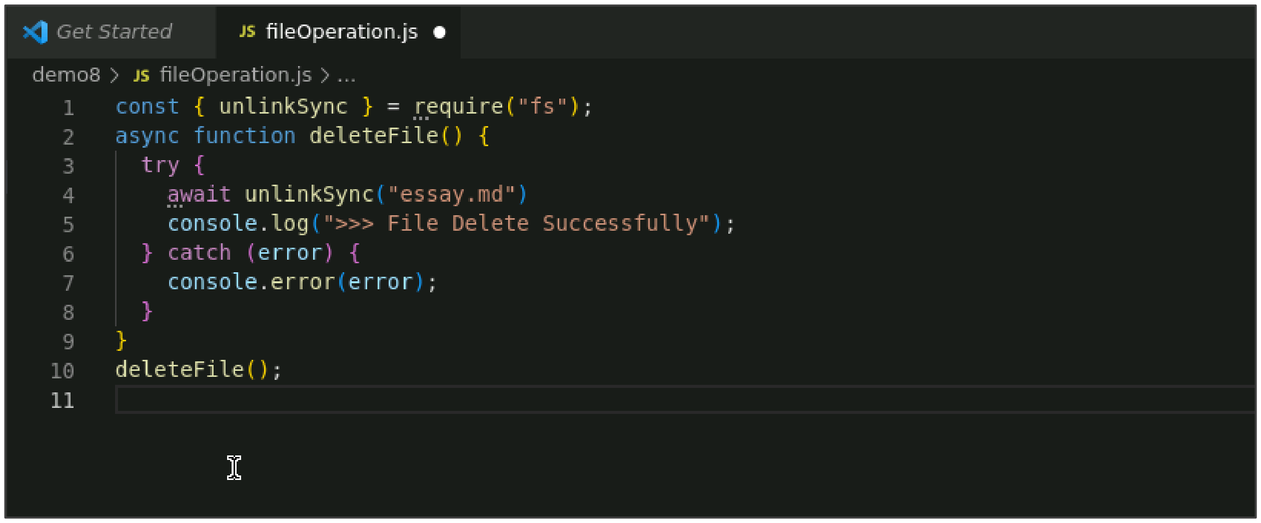


demo8 > JS fileOperation.js > ...

```
const { unlink } = require("fs");

const { unlink } = require("fs");

// Non-Blocking Task
unlink("./essay.md", (error) => {
  if (error) {
    console.error(error);
    return;
  }

  console.log(">>> File Delete Successfully");
})

```

3.3 Use the following code to delete the file synchronously:

```
const { unlinkSync } = require("fs");
async function deleteFile() {
  try {
    await unlinkSync("essay.md")
    console.log(">>> File Delete Successfully");
  } catch (error) {
    console.error(error);
  }
}
deleteFile();
```



```
const { unlinkSync } = require("fs");
async function deleteFile() {
  try {
    await unlinkSync("essay.md")
    console.log(">>> File Delete Successfully");
  } catch (error) {
    console.error(error);
  }
}
deleteFile();
```

By following these steps, you have successfully executed file system operations like reading, writing, and deleting to manage files within a file system.