Date: 8 June 2024 **Day: Saturday**

Overview:

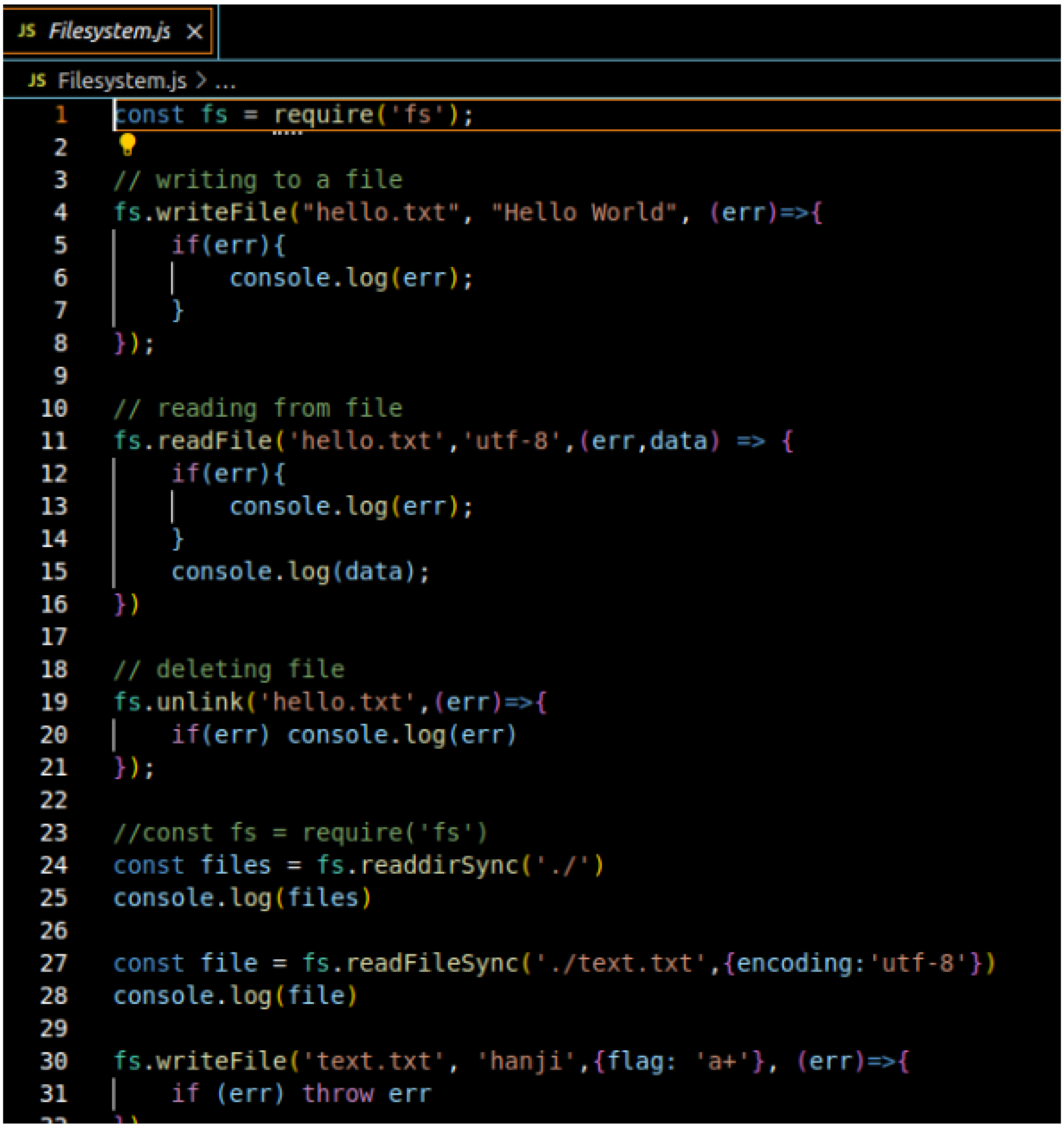
Day 4 of the internship focused on practical implementations of file system operations in Node.js. This included synchronous and asynchronous file reading, writing data to files, and appending content. These operations are essential for handling data persistence and manipulation in applications.

Learning Objectives:

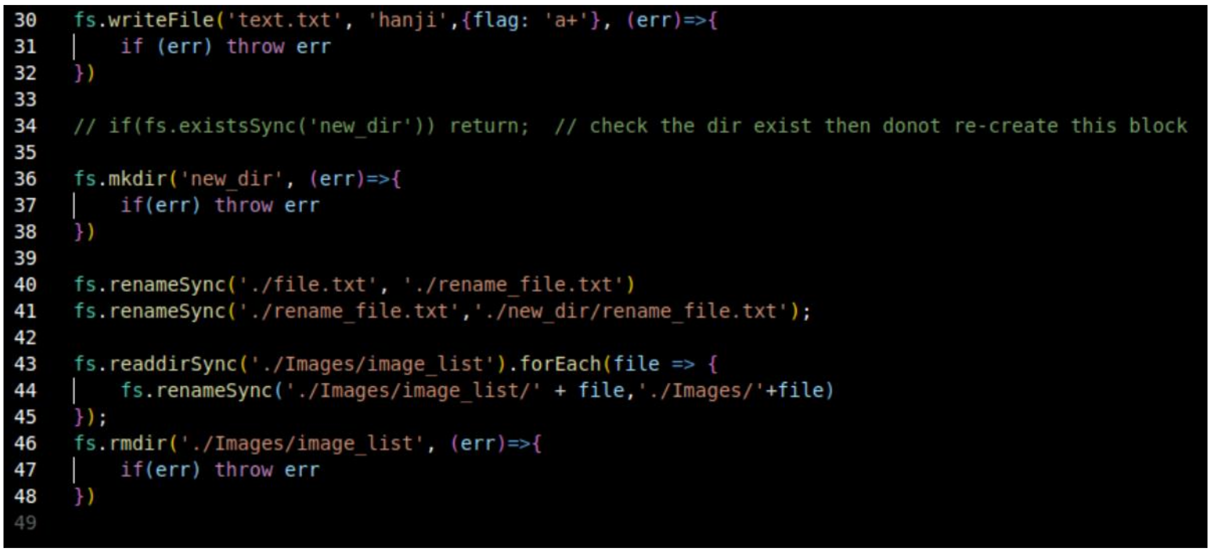
- File System Operations in Node.js:
- Explored core functionalities for reading, writing, and appending files in Node.js.
- Learned techniques for handling file I/O efficiently using synchronous and asynchronous methods.

Reading Files:

- Synchronous Reading: Implemented synchronous file reading operations using fs.readFileSync(). This method blocks further execution until the file is completely read, suitable for smaller files and straightforward workflows.
- Asynchronous Reading: Practiced asynchronous file reading with fs.readFile() to avoid blocking the main thread, ensuring application responsiveness during file operations.



```
const fs = require('fs');

// writing to a file
fs.writeFile("hello.txt", "Hello World", (err)=>{
    if(err){
        console.log(err);
    }
});

// reading from file
fs.readFile('hello.txt','utf-8',(err,data) => {
    if(err){
        console.log(err);
    }
    console.log(data);
})

// deleting file
fs.unlink('hello.txt',(err)=>{
    if(err) console.log(err)
});

//const fs = require('fs')
const files = fs.readdirSync('./')
console.log(files)

const file = fs.readFileSync('./text.txt',{encoding:'utf-8'})
console.log(file)

fs.writeFile('text.txt', 'hanji',{flag: 'a+'}, (err)=>{
    if (err) throw err
```



```
fs.writeFile('text.txt', 'hanji',{flag: 'a+'}, (err)=>{
    if (err) throw err
})

// if(fs.existsSync('new_dir')) return;  // check the dir exist then donot re-create this block

fs.mkdir('new_dir', (err)=>{
    if(err) throw err
})

fs.renameSync('./file.txt', './rename_file.txt')
fs.renameSync('./rename_file.txt','./new_dir/rename_file.txt');

fs.readdirSync('./Images/image_list').forEach(file => {
    fs.renameSync('./Images/image_list/' + file,'./Images/'+file)
});
fs.rmdir('./Images/image_list', (err)=>{
    if(err) throw err
})
```

Writing and Appending Files:

- **File Writing:** Implemented writing data to files using both synchronous (fs.writeFileSync()) and asynchronous (fs.writeFile()) methods. These operations allow for data persistence and content creation within applications.
- **Appending Data:** Explored appending new content to existing files using fs.appendFile(). This method is beneficial for logging and incremental data updates without overwriting existing information.

Activities and Insights:

- Implementation of File System Operations: Actively implemented each file system operation to solidify understanding and practical application. This included handling edge cases such as file not found errors and permission issues.
- Comparison of Synchronous vs. Asynchronous Methods: Compared the advantages and trade-offs between synchronous and asynchronous file operations. Emphasized the importance of asynchronous methods in non-blocking I/O scenarios to maintain application performance.
- Error Handling and Best Practices: Implemented robust error handling strategies using try-catch blocks for synchronous operations and error-first callback functions for asynchronous operations. Ensured comprehensive error management to enhance application reliability.