

# File Operations with Logging

JOLIANA EMAD KAMAL NAGUIB 20P3292

# TASK Node.js

## File Operations with Logging:

Write a Node.js script that reads the contents of a file asynchronously, modifies the contents by appending a timestamp, and writes the updated contents to a new file.

Ensure the file operations are performed asynchronously using the `fs` module.

Logging:

Use the `winston` module to log the file operations. Each time a file is read, written, or an error occurs, log a message with the details of the operation.

Rotate the log files on a daily basis using the `winston-daily-rotate-file` module.

Steps:

Install necessary modules: `npm install winston winston-daily-rotate-file fs`.

Create a function that reads a file asynchronously (`readFile`), appends a timestamp, and writes it to a new file (`writeFile`).

Implement logging using the `winston` module to log the file read, write, and any errors.

Ensure log rotation happens daily.

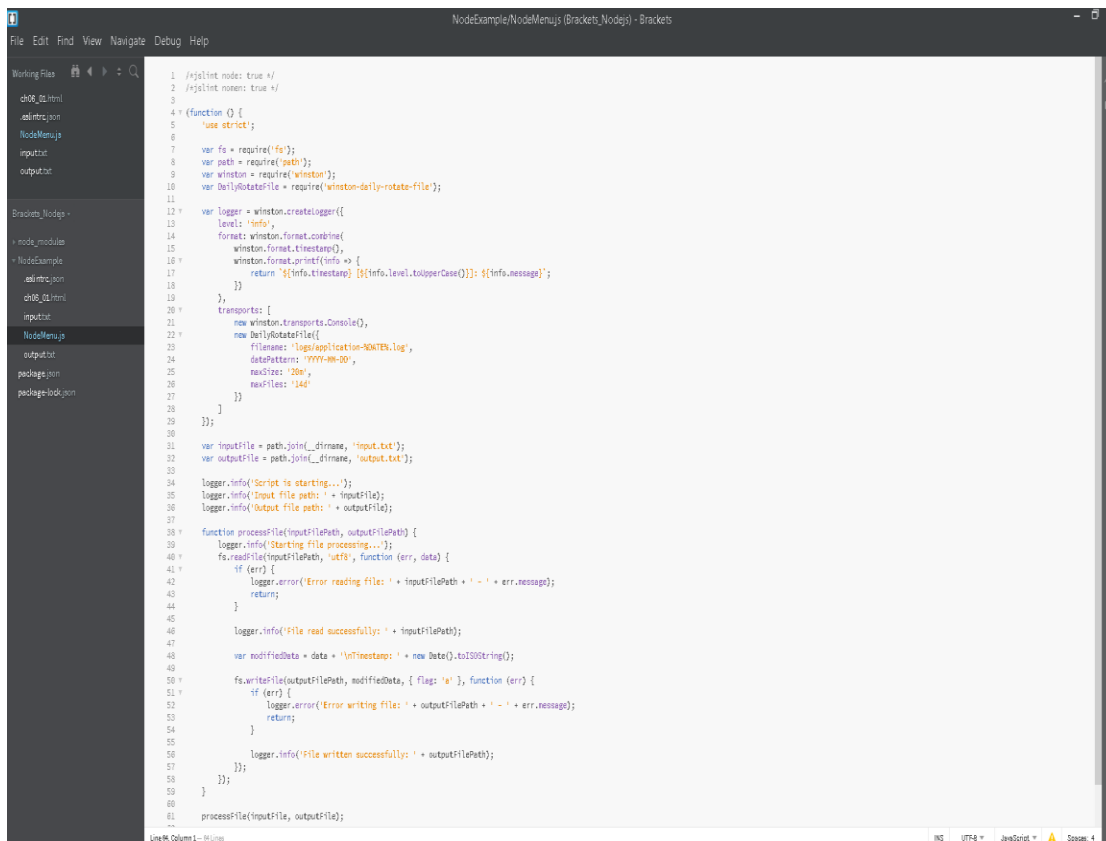
Expected Outcome:

A Node.js script that performs the file operations and logs the process with daily log rotation.

This task involves working with file operations and implementing logging, combining concepts from both sections.

## NodeMenu.js

->NodeMenu.js script inside NodeExample folder that reads the contents of a file asynchronously, modifies the contents by appending a timestamp and writes updated to a new file.



```
1 //eslint node: true */
2 //eslint nomen: true */
3
4 + (function () {
5     'use strict';
6
7     var fs = require('fs');
8     var path = require('path');
9     var winston = require('winston');
10    var DailyRotateFile = require('winston-daily-rotate-file');
11
12 +    var logger = winston.createLogger({
13        level: 'info',
14        format: winston.format.combine(
15            winston.format.timestamp(),
16            winston.format.printf(function (info) {
17                return '[${info.timestamp}] ${info.level.toUpperCase()} ${info.message}';
18            })
19        ),
20        transports: [
21            new winston.transports.Console(),
22            new DailyRotateFile({
23                filename: 'logs/application-NODEJS.log',
24                datePattern: 'YYYY-MM-DD',
25                maxSize: '20m',
26                maxFiles: '14d'
27            })
28        ]
29    });
30
31    var inputFile = path.join(__dirname, 'input.txt');
32    var outputFile = path.join(__dirname, 'output.txt');
33
34    logger.info('Script is starting...');
35    logger.info('Input file path: ' + inputFile);
36    logger.info('Output file path: ' + outputFile);
37
38 +    function processFile(inputFilePath, outputFilePath) {
39        logger.info('Starting file processing...');
40        fs.readFile(inputFilePath, 'utf8', function (err, data) {
41            if (err) {
42                logger.error('Error reading file: ' + inputFile + ' - ' + err.message);
43                return;
44            }
45
46            logger.info('File read successfully: ' + inputFile);
47
48            var modifiedData = data + '\nTimestamp: ' + new Date().toLocaleString();
49
50            fs.writeFile(outputFilePath, modifiedData, { flags: 'a' }, function (err) {
51                if (err) {
52                    logger.error('Error writing file: ' + outputFile + ' - ' + err.message);
53                    return;
54                }
55
56                logger.info('File written successfully: ' + outputFile);
57            });
58        });
59    }
60
61    processFile(inputFile, outputFile);
```

```

1  /*eslint node: true */
2  /*eslint nomen: true */
3
4  ▼ (function () {
5      'use strict';
6
7      var fs = require('fs');
8      var path = require('path');
9      var winston = require('winston');
10     var DailyRotateFile = require('winston-daily-rotate-file');
11
12     ▼ var logger = winston.createLogger({
13         level: 'info',
14         format: winston.format.combine(
15             winston.format.timestamp(),
16             winston.format.printf(info => {
17                 return `${info.timestamp} [${info.level.toUpperCase()}]: ${info.message}`;
18             })
19         ),
20         ▼ transports: [
21             new winston.transports.Console(),
22             ▼ new DailyRotateFile({
23                 filename: 'logs/application-%DATE%.log',
24                 datePattern: 'YYYY-MM-DD',
25                 maxSize: '20m',
26                 maxFiles: '14d'
27             })
28         ]

```

```

26         maxFiles: '14d'
27     })
28 ]
29 });
30
31 var inputFile = path.join(__dirname, 'input.txt');
32 var outputFile = path.join(__dirname, 'output.txt');
33
34 logger.info('Script is starting...');
35 logger.info('Input file path: ' + inputFile);
36 logger.info('Output file path: ' + outputFile);
37
38 ▼ function processFile(inputFilePath, outputFilePath) {
39     logger.info('Starting file processing...');
40     ▼ fs.readFile(inputFilePath, 'utf8', function (err, data) {
41         ▼ if (err) {
42             logger.error('Error reading file: ' + inputFilePath + ' - ' + err.message);
43             return;
44         }
45
46         logger.info('File read successfully: ' + inputFilePath);
47
48         var modifiedData = data + '\nTimestamp: ' + new Date().toISOString();
49
50     ▼ fs.writeFile(outputFilePath, modifiedData, { flag: 'a' }, function (err) {
51         ▼ if (err) {
52             logger.error('Error writing file: ' + outputFilePath + ' - ' + err.message);
53             return;
54         }
55
56         logger.info('File written successfully: ' + outputFilePath);
57     });
58 }
59
60 processFile(inputFile, outputFile);

```

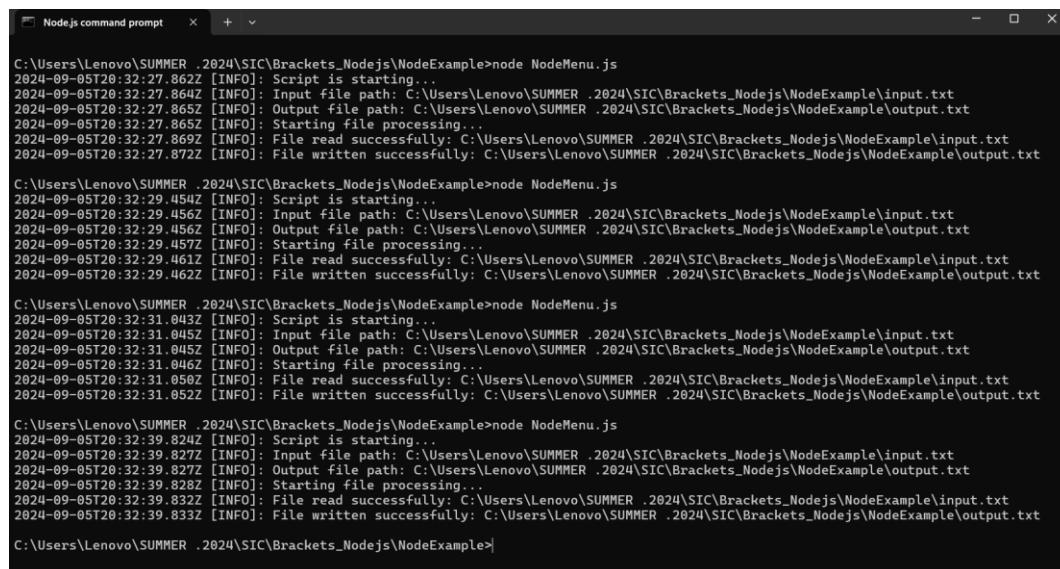
## ESLint configuration file



```
1 {
2   "env": {
3     "node": true,
4     "es6": true
5   },
6   "parserOptions": {
7     "ecmaVersion": 2017
8   },
9   "rules": {
10    "no-console": "off",
11    "no-undef": "off", // This turns off the rule flagging 'require' and '__dirname' as undefined
12    "no-underscore-dangle": "off" // Allows dangling underscores like in '__dirname'
13  }
14 }
15
16 |
```

## Node.js Terminal

➔ Use Node NodeMenu.js to run file



```
Node.js command prompt
C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample>node NodeMenu.js
2024-09-05T20:32:27.862Z [INFO]: Script is starting...
2024-09-05T20:32:27.864Z [INFO]: Input file path: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\input.txt
2024-09-05T20:32:27.865Z [INFO]: Output file path: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\output.txt
2024-09-05T20:32:27.865Z [INFO]: Starting file processing...
2024-09-05T20:32:27.869Z [INFO]: File read successfully: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\input.txt
2024-09-05T20:32:27.872Z [INFO]: File written successfully: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\output.txt

C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample>node NodeMenu.js
2024-09-05T20:32:29.454Z [INFO]: Script is starting...
2024-09-05T20:32:29.456Z [INFO]: Input file path: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\input.txt
2024-09-05T20:32:29.456Z [INFO]: Output file path: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\output.txt
2024-09-05T20:32:29.457Z [INFO]: Starting file processing...
2024-09-05T20:32:29.461Z [INFO]: File read successfully: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\input.txt
2024-09-05T20:32:29.462Z [INFO]: File written successfully: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\output.txt

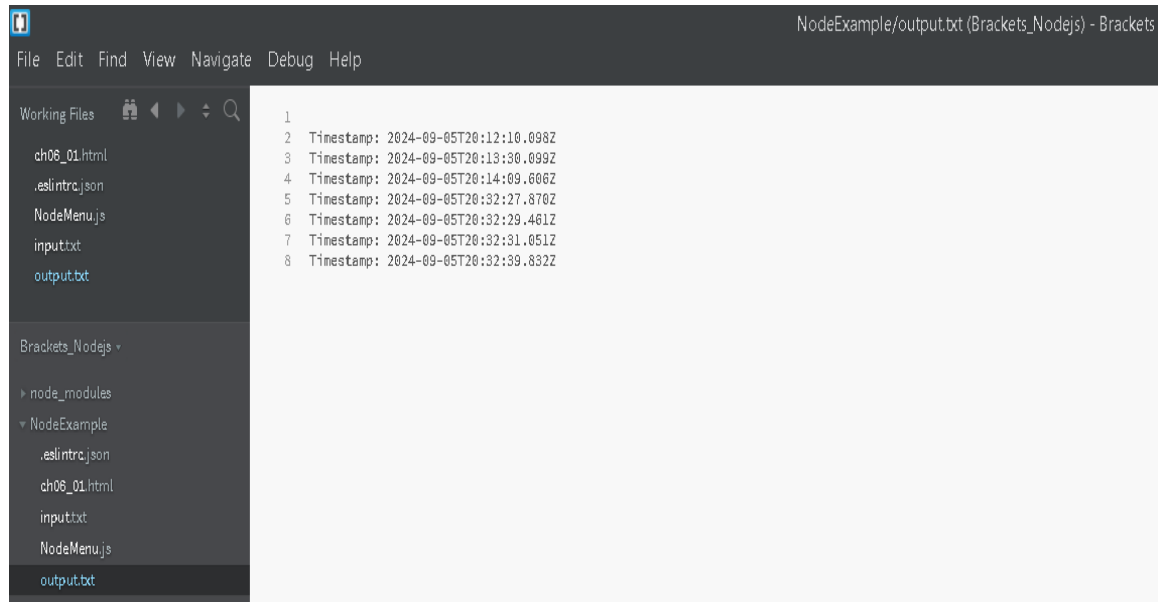
C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample>node NodeMenu.js
2024-09-05T20:32:31.043Z [INFO]: Script is starting...
2024-09-05T20:32:31.045Z [INFO]: Input file path: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\input.txt
2024-09-05T20:32:31.045Z [INFO]: Output file path: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\output.txt
2024-09-05T20:32:31.046Z [INFO]: Starting file processing...
2024-09-05T20:32:31.050Z [INFO]: File read successfully: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\input.txt
2024-09-05T20:32:31.052Z [INFO]: File written successfully: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\output.txt

C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample>node NodeMenu.js
2024-09-05T20:32:39.824Z [INFO]: Script is starting...
2024-09-05T20:32:39.827Z [INFO]: Input file path: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\input.txt
2024-09-05T20:32:39.827Z [INFO]: Output file path: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\output.txt
2024-09-05T20:32:39.828Z [INFO]: Starting file processing...
2024-09-05T20:32:39.832Z [INFO]: File read successfully: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\input.txt
2024-09-05T20:32:39.833Z [INFO]: File written successfully: C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample\output.txt

C:\Users\Lenovo\SUMMER .2024\SIC\Brackets_Nodejs\NodeExample>
```

## Output text file

➔ Output file after logs the process with daily log rotation



The screenshot shows the Brackets code editor interface. The title bar at the top right reads "NodeExample/output.txt (Brackets\_Nodejs) - Brackets". The menu bar includes "File", "Edit", "Find", "View", "Navigate", "Debug", and "Help". The left sidebar, titled "Working Files", contains a list of files: "ch06\_01.html", ".eslintrc.json", "NodeMenu.js", "input.txt", and "output.txt", with "output.txt" selected. Below this, a tree view shows the project structure: "Brackets\_Nodejs" (expanded) containing "node\_modules", "NodeExample" (expanded), and its sub-files: ".eslintrc.json", "ch06\_01.html", "input.txt", "NodeMenu.js", and "output.txt". The main editor area displays the content of "output.txt", which consists of eight lines of timestamps, each preceded by a line number from 1 to 8.

```
1
2 Timestamp: 2024-09-05T20:12:10.098Z
3 Timestamp: 2024-09-05T20:13:30.099Z
4 Timestamp: 2024-09-05T20:14:09.606Z
5 Timestamp: 2024-09-05T20:32:27.870Z
6 Timestamp: 2024-09-05T20:32:29.461Z
7 Timestamp: 2024-09-05T20:32:31.051Z
8 Timestamp: 2024-09-05T20:32:39.832Z
```