Node.js fs Module

Mohamed Mukthar Ahmed

1

Node.js fs Module

Outline

- The File System (fs) Module
  - Synchronous v/s Asynchronous
- Common Methods
- Reading from a file
- Writing to a file
- Concerns
- Other Methods
- Directory Manipulation
- Miscellaneous Methods

Mohamed Mukthar Ahmed

2

Node.js fs Module

- Functionality to access and interact with the file system
- Helps in handling file operations like – reading, creating, deleting etc.
- Node.js gives the functionality of file I/O by providing wrappers around the standard POSIX functions.
- All file system operations can have **synchronous** and **asynchronous** forms depending upon user requirements.

3

Synchronous vs Asynchronous

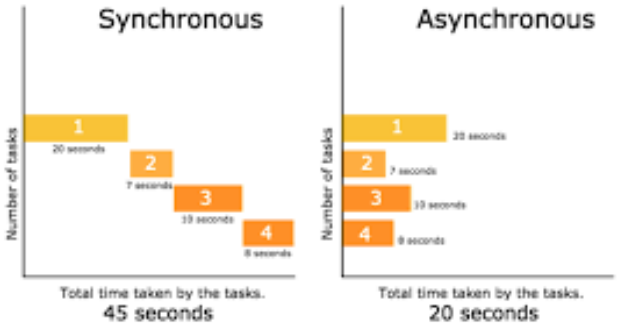
ANALOGY

- **Synchronous**
  - You are in a queue to get a movie ticket. You cannot get one until everybody in front of you gets one, and the same applies to the people queued behind you.
- **Asynchronous**
  - You are in a restaurant with many other people. You order your food. Other people can also order their food, they don't have to wait for your food to be cooked and served to you before they can order. In the kitchen restaurant workers are continuously cooking, serving, and taking orders. People will get their food served as soon as it is cooked.

4

Synchronous vs Asynchronous

ADVANTAGE



5

Node.js fs Module

- All file system operations can have **synchronous** and **asynchronous** forms depending upon user requirements.

Method	Description
`fs.readFile`	Read the contents of a file
`fs.writeFile`	Write contents to the file
`fs.readFileSync`	Read the contents of the file synchronously
`fs.writeFileSync`	Write the contents of the file synchronously

6

Reading Files-1

- Use the **fs.readFile()** method, passing it the file path, encoding and a callback function that will be called with the file data (and the error)

```
fs.readFile('./assets/file1.txt', 'utf-8',(err, data) => {
    if (err) {
        console.error(err);
        return;
    }
    else
        console.log(data);
})
```


7

Reading Files-2

- Alternatively, you can use the synchronous version **fs.readFileSync()**

```
try {
    const first = fs.readFileSync('./assets/file1.txt', 'utf-8');
    console.log(first);
} catch (err) {
    console.error(err);
}
```


8

Writing Files – 1

- To write to files to use the **fs.writeFile()** API

```
const content = 'Some content!';

fs.writeFile('./assets/file3.txt', content, err => {
  if (err) {
    console.error(err);
  }
  // file written successfully
});
```

- Alternatively, use synchronous **fs.writeFileSync()** API

```
const content = 'Some content!';

try {
  fs.writeFileSync('./assets/file3.txt', content);
  // file written successfully
} catch (err) {
  console.error(err);
}
```


9

Writing Files – 2

- By default, the APIs will replace the contents of the file if it does already exist.
- You can modify the default by specifying a **flag**

```
fs.writeFile('./assets/file3.txt', content, { flag: 'a+' }, err => {});
```

- A handy method to **append** content to the end of a file is **fs.appendFile()** (and its **fs.appendFileSync()** counterpart):

```
fs.appendFile('file.log', content, err => {
  if (err) {
    console.error(err);
  }
  // done!
});
```


10

Concerns

- Both **fs.readFile()** and **fs.readFileSync()** read the full content of the file in memory before returning the data.
- Both **fs.writeFile()** and **fs.WriteFileSync()** methods write the full content to the file before returning the control back to your program.
- In the async version, this means executing the callback
- This means that **big files** are going to have a major impact on the **memory consumption** and **speed of execution** of the program.
- A better option is to read and write file content using **streams**.

11

Reading & Writing files

- Other methods for reading and writing the conventional way are:

Method	Description
`fs.open`	Open a file for create, read or write
`fs.read`	Read file specified by FD
`fs.write`	Write to file specified by FD
`fs.close`	Close the opened file

12

Directory Manipulation

- Methods for making, reading and removing directory

Method	Description
`fs.mkdir`	Make a directory
`fs.mkdirSync`	
`fs.readdir`	Read a directory
`Fs.readdirSync`	
`fs.rmdir`	Remove a directory
`fs.rmdirSync`	

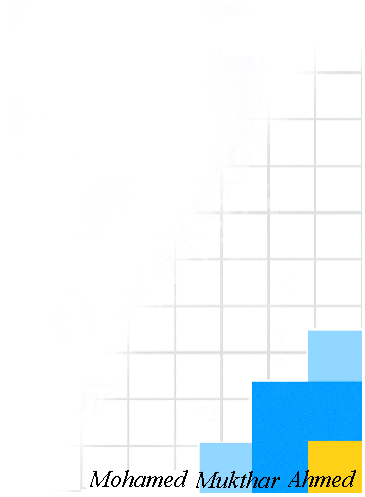
13

Miscellaneous Methods

- We have a number of methods in the Node.js fs module

Method	Description
`fs.exits`	Checks for the existence of a file
`fs.existsSync`	
`fs.rename`	Rename a file
`Fs.renameSync`	
`fs.fstat`	File status information
`fs.fstatSync`	
`fs.unlink`	Remove a file
`fs.unlinkSync`	
`fs.truncate`	
`fs.truncateSync`	

14



15