Week-1

Backend Engineering Launchpad

Pawan Panjwani

Airtribe

Backend Development Fundamentals of Node JS

Airtribe

About me

Hello, I'm Pawan Panjwani

With 4+ years of experience, I have been in the software engineering space for both startups and multinational product based companies. Throughout my experience, I have worked on scalable, reliable and complex distributed systems

LinkedIn BrowserStack

Airtribe

Agenda

- Introduction to Node.js
- Why use Node.js ?
- Understanding asynchronous nature of Node.js
- Understanding Package Manager, Importing and exporting the modules
- Understanding package.json
- Live coding - File handling in Node.js
- Question and answers

What is Node JS

- Node Js is an asynchronous javascript runtime build on the chrome's v8 runtime engine.
- Node js Uses an event driven, non blocking I/O model that makes it light weight and efficient
- **Node Js is single threaded.**
- In simple words, Node js is **server sided javascript.**

Why Node JS

- Lightweight and very fast !!
- Asynchronous and event driven - Many connections can be handled concurrently !

Where to use Node JS?

- I/O Bound applications
- Data streaming applications
- JSON API based applications

NOTE: It is not advisable to use node js for CPU intensive applications (Think of CPU intensive as infinite for loop which blocks the main thread or performs some huge computations on the main thread)

Blocking vs non-blocking

- **Blocking** methods execute synchronously
- **Non blocking** methods execute asynchronously
- **Blocking** is when the execution of the code waits until the currently running operation completes.
- **Non blocking** is when the execution of the code does more work, until the currently running operation completes

Blocking

Code waits on line 2 until the data is read from the file, and moves to line 3 only when the reading the file is completed.

Airtribe

Non-blocking

Code does not wait for the file read to complete and console to be logged, it calls moreWork() even before the console.log statement is called. This is being achieved using callbacks which are a way to write asynchronous code.

Airtribe

Some Theory: Blocking vs Non-blocking IO

```
1   // Traditional, Blocking IO //
2
3   var result = db.query("select * from some_table");
4   doSomethingWithResult(result); // wait for result!
5   doSomethingWithoutResult(); // Execution is blocked
6
7
8   // Non Traditional, Non Blocking IO
9
10  db.query("select * from some_table", function(result) {
11    doSomethingWithResult(result); // wait for result inside the callback
12  });
13  doSomethingWithoutResult(); // executes without any delay and without being blocked
14
15
```

You can do something without result in non blocking IO node js, without being blocked on the IO operation, what makes node JS achieve this ?

Event Loop and event driven execution

Airtribe

Node package manager

- Node Js supports modularity. your node js application can contain third party modules each of which can perform a specific functionality (For ex: fs for I/O file operations.)
- NPM is the node package manager used to install, update, uninstall, configure node js platform modules/packages easily.
- Npm install commands: npm install --save fs

Where are packages and modules tracked ?

- Modules and the packages of your node js application are tracked inside package.json file.
- Package.json is a JSON file that lives in the root directory of your project.
- Your *package.json* holds important information about the project. It contains human-readable metadata about the project (like the project name and description) as well as functional metadata like the package version number and a list of dependencies required by the application.

Structure of the package .json

```
{
  "name": "airtribe-course-rating-app",
  "version": "1.0.0",
  "description": "airtribe course rating app",
  "main": "app.js",
  ▷ Debug
  "scripts": {
    "start": "nodemon src/app.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.2",
    "cors": "^3.8.6",
    "express": "^4.18.2"
  },
  "devDependencies": {
    "nodemon": "^2.0.20"
  }
}
```

Airtribe

Installing third party modules in Node Js

- Npm Install
  - Npm install <options> <package unique name>
- Example: npm install –save express
- The modules/packages available can be imported in .js file using

  “require” function.

  Example: require(“express”)

  This helps in loading the “express” package in the current .js file

Modules export and import

- Modules in node JS can also be exported, so that their functionality can be imported in other `.js` files
- “Require” function helps in importing the modules and “module.exports” is used for exporting the modules
- **Exporting -**
  - module.exports{<module to be exported>}
- **Importing -**
  - var <varName> = require(<module name>)

Lets create Airtribe File Manager

Associated session - session #1 (introduction to Node Js and NPM)

Write a simple script named *airtribe-file-manager (KFM)* in node js, the script would be responsible for reading the file from one directory and writing it to another directory.

Source folder name: *AFM-source*
Destination folder name: *AFM-destination*
Name of the file: input.txt
Contents of the file are as follows:

“I love airtribe launchpad backend development”

Lets create Airtribe File Manager

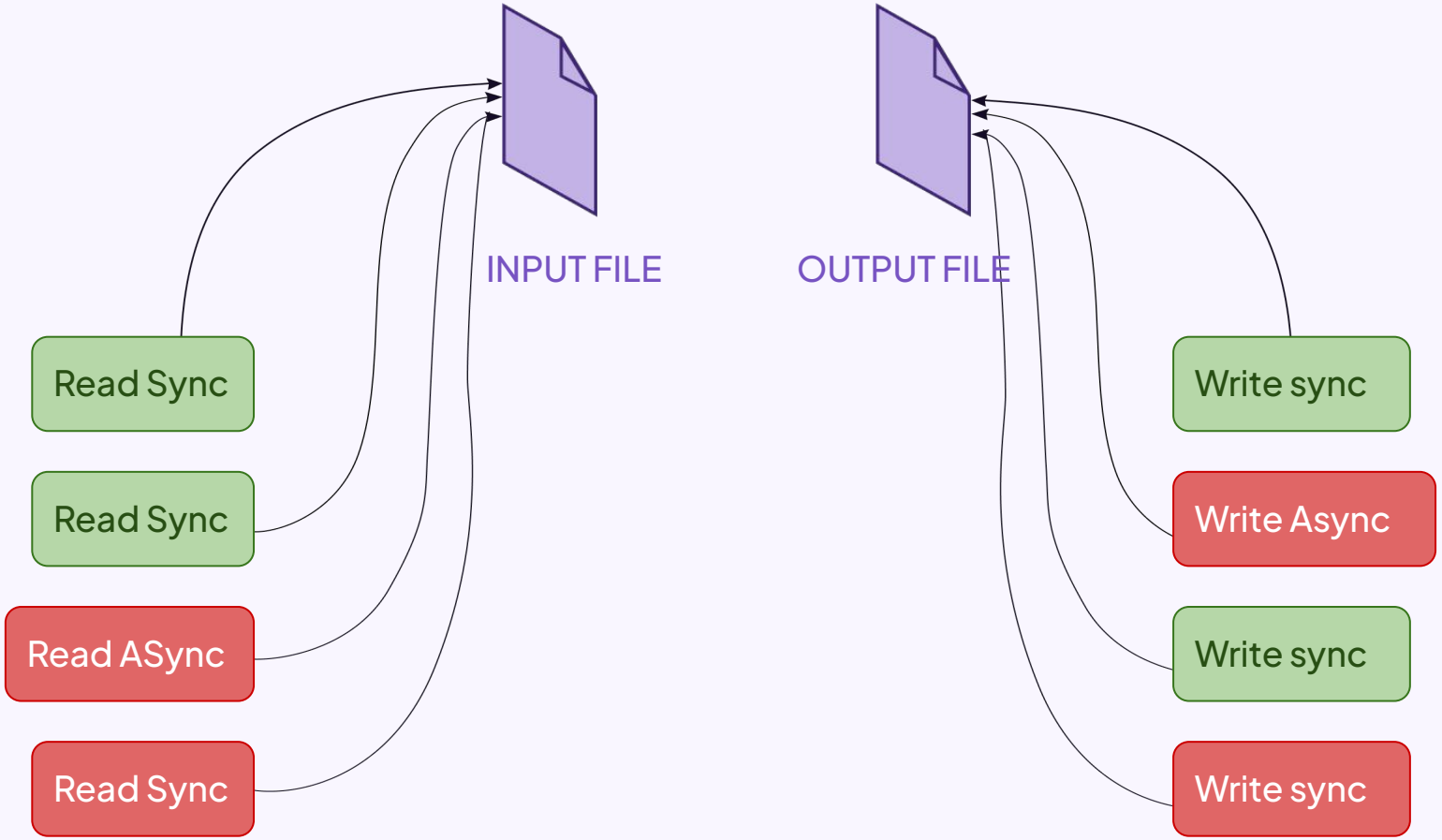
Your application should be able to perform these operations in the following manner

- Read and write the file synchronously
- Read and write the file asynchronously
- Read the file synchronously and write the file asynchronously
- Read the file asynchronously and write the file synchronously

Lets create Airtribe File Manager

Outcomes:

- Writing a simple script in node js
- Understanding the file handling in node js
- Understanding the blocking and non blocking operations in node js
- Understanding the core fs module of the node js
- Understanding I/O operations and why node js can be a better fit for it


INPUT FILE
OUTPUT FILE
Read Sync
Read Sync
Read ASync
Read Sync
Write sync
Write Async
Write sync
Write sync

Airtribe

Understanding Node JS with synchronous/asynchronous IO operations

1. https://github.com/airtribe-projects/bel-p1/tree/main/airtribe-file-manager

30-Min Live Exercise

Thank you

Airtribe