Date: 14 June 2024 Day: Friday

Overview:

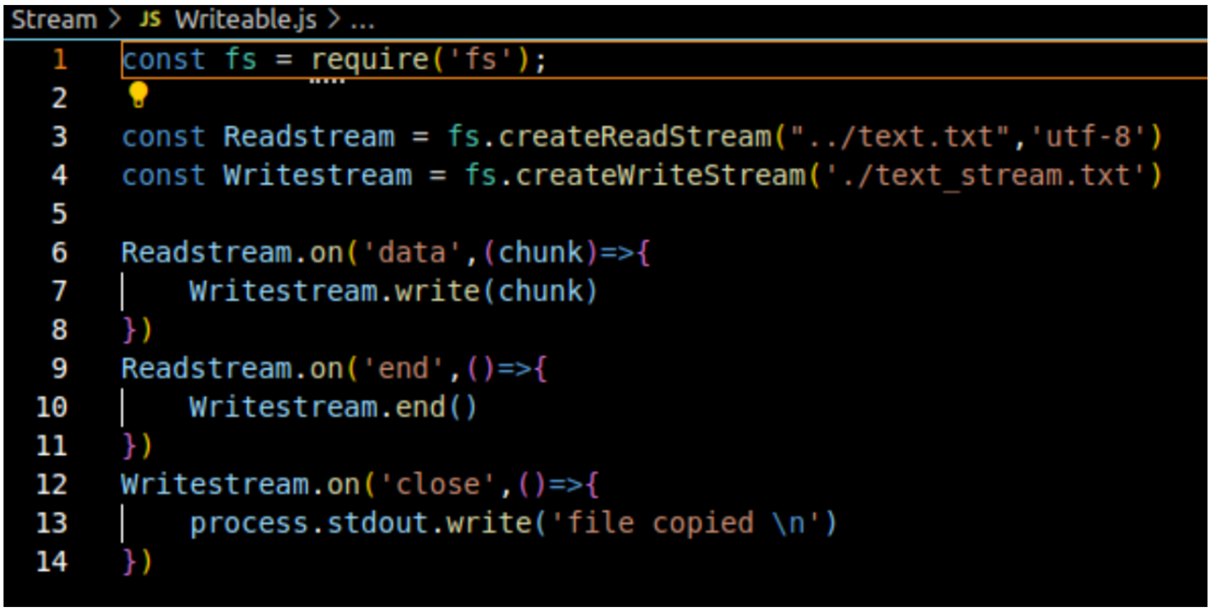
Day 8 of the internship focused on understanding and utilizing npm (Node Package Manager), a crucial tool for managing dependencies and packages in Node.js applications. npm facilitates easy integration of third-party libraries, modules, and tools, enhancing the functionality and scalability of Node.js projects.

Learning Objectives:

- **Understanding npm:**
  - Explored the significance of npm in the Node.js ecosystem as a package manager for JavaScript.
  - Learned how npm simplifies the process of discovering, installing, and managing packages and dependencies for Node.js applications.
- **Dependencies vs. devDependencies:**
  - Differentiated between dependencies and devDependencies in npm:
  - Dependencies: Essential packages required for the application to run in production environments. These are typically runtime dependencies.
  - devDependencies: Packages needed during development, such as testing frameworks, build tools, and code quality

tools. These are not necessary for the production runtime environment.

- **Using npm Scripts:**
    - Practiced creating and using npm scripts to automate common development tasks:
    - Custom Scripts: Defined custom scripts in package.json for tasks such as running tests, starting the development server, linting code, and building production bundles.
    - Script Execution: Executed npm scripts using npm run <script-name> to streamline development workflows and enhance productivity.
    - Chaining Scripts: Chained multiple scripts together using && or npm-run-all for sequential or parallel execution of tasks.

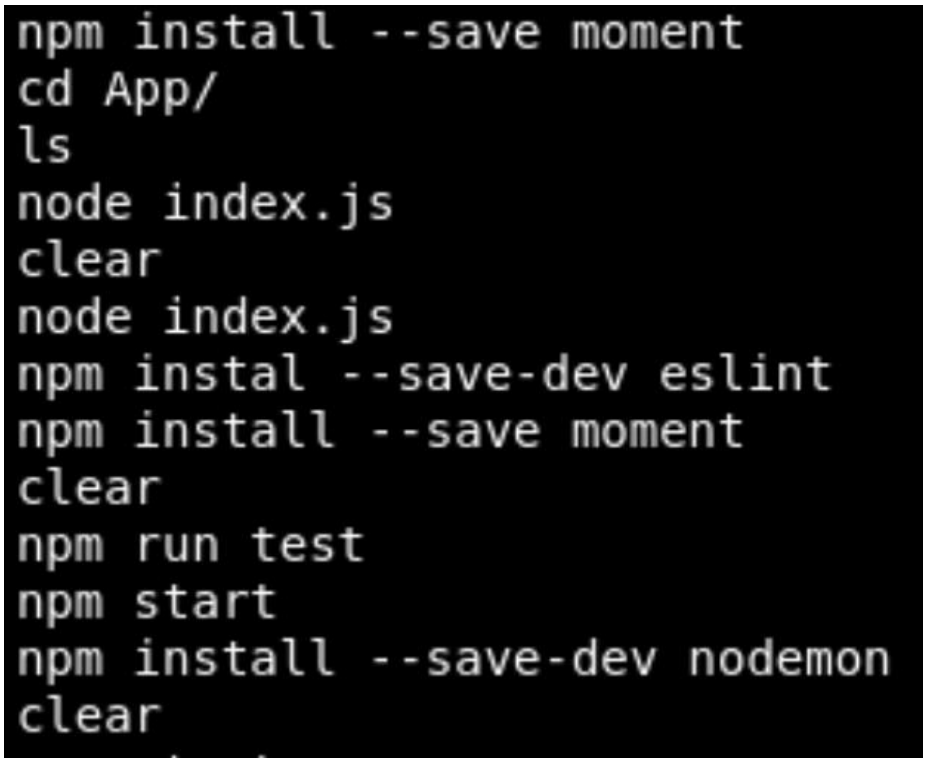

Stream > JS Writeable.js > ...

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

const Readstream = fs.createReadStream("../text.txt",'utf-8')
const Writestream = fs.createWriteStream('./text_stream.txt')

Readstream.on('data',(chunk)=>{
    Writestream.write(chunk)
})
Readstream.on('end',()=>{
    Writestream.end()
})
Writestream.on('close',()=>{
    process.stdout.write('file copied \n')
})
```

- **npm Commands and Operations:**

Covered essential npm commands for package management:

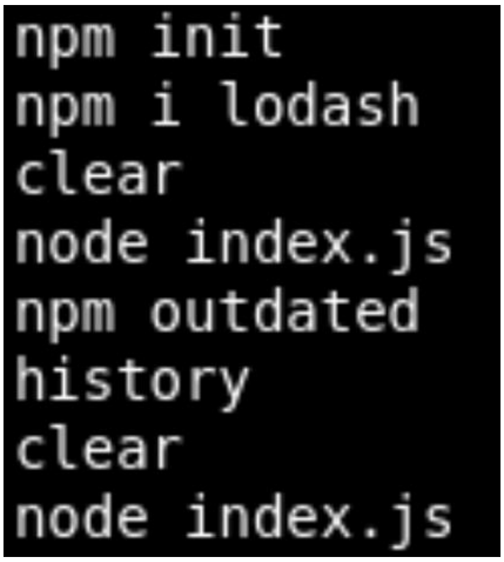
- npm install: Used for installing dependencies listed in package.json.
- npm update: Updates packages to their latest versions based on specified criteria.
- npm uninstall: Removes installed packages from the project
- npm init: Initializes a new package.json file to manage project metadata and dependencies.



```
npm install --save moment
cd App/
ls
node index.js
clear
node index.js
npm instal --save-dev eslint
npm install --save moment
clear
npm run test
npm start
npm install --save-dev nodemon
clear
```

- W**orking with External npm Modules:**
    - Discussed strategies for evaluating and selecting external npm modules to extend Node.js applications.
    - Explored common npm modules and their use cases, such as Nodemon for real-time changes, Lodash for utility functions.

    - Integrated external npm modules into sample projects to demonstrate functionality and interoperability.



```
npm init
npm i lodash
clear
node index.js
npm outdated
history
clear
node index.js
```

Activities and Insights:

- **Practical Use of npm Commands:**
    - Executed npm commands to install, update, and remove packages, managing project dependencies efficiently.
    - Initiated new projects with npm init to establish package.json files, configuring project metadata and dependencies.

- **Documentation and Best Practices:**
    - Documented npm installation procedures, dependencies, and versioning strategies to facilitate collaboration and future maintenance.
    - Discussed best practices for dependency management, including version pinning, semantic versioning, and security considerations.