Read File, Write File, .env, Server Organization

Software Development Bootcamp

Topic

fs module, read & write file

What Is the fs Module?

- The `fs` module in Node.js provides a way of working with the file system on your computer
- It allows you to read from and write to files, as well as perform other file-related operations
- `fs` stands for “file system”

`fs` Module Key Points

- Part of Node.js core modules (no installation required)
- Provides both synchronous and asynchronous methods
- Supports operations like reading, writing, updating, and deleting files
- Essential for server-side file management in Node.js applications

What Is `fs.readFile`?

`fs.readFile` is an asynchronous function used to read the entire contents of a file

- **Asynchronous**: Doesn't block the execution of other code
- **Flexible**: Can read various file types (text, binary, etc.)
- **Error-First Callback**: Follows Node.js conventions for handling errors

What Is `fs.writeFile`?

fs.writeFile is an asynchronous function used to write data to a file, replacing the file if it already exists

- **Asynchronous**: Doesn't block the execution of other code
- **Creates or Overwrites**: Will create a new file if it doesn't exist, or overwrite if it does
- **Flexible**: Can write various data types (strings, buffers, etc.)
- **Options**: Allows specifying file mode, encoding, etc.

Error Handling

- Both **readFile** and **writeFile** use error-first callbacks
- Always check for errors before processing the result
- Common errors:
  - File not found
  - Permission denied
  - Disk full (for write operations)

Why Use `readFile` and `writeFile`?

- **`fs.readFile`** and **`fs.writeFile`** are powerful tools for file operations in Node.js
- They enable your application to interact with the file system efficiently
- Understanding these methods is crucial for backend development with Node.js

Exercise

Hello Files

Topic

.env

What Is A `.env` File?

- A `.env` file is a configuration file used to store environment variables
- It's a simple text file with key-value pairs
- Commonly used to store sensitive information like API keys, database credentials, etc.

Why Use `.env` File With Express?

- **Security**: Keep sensitive data out of your codebase
- **Flexibility**: Easily change configuration between environments (development, staging, production)
- **Best Practice**: Follows the Twelve-Factor App methodology for config management

Best Practices for `.env` Files

- `.env` always goes inside `.gitIgnore` never commit .env files to GitHub
- Include a `.env.example` file with dummy values in your repository
- Use different `.env` files for different environments
- Validate required environment variables on app startup

Setting Up `.env` In Postman

- Create a new environment in Postman
  - Click on the "Environments" tab
  - Click "New" to create a new environment
- Add variables to your environment
  - Set variable names and their corresponding values
  - Example: `PORT: 3000`
- Save your environment

Topic

JSON As A Mock Database

Why Use JSON Files As A Mock Database?

- Quick setup for prototyping
- No need for a separate database server
- Easy to version control
- Suitable for small-scale applications or testing
- Allows focus on API design before implementing a full database

Structure Of A JSON Mock Database

- Create a `.json` file (e.g. `db.json`)
- Structure data as an object with arrays

```
{
  "users": [
    {"id": 1, "name": "Alice", "email": "alice@example.com"},
    {"id": 2, "name": "Bob", "email": "bob@example.com"}
  ],
  "posts": [
    {"id": 1, "userId": 1, "title": "First Post", "body": "..."},
    {"id": 2, "userId": 2, "title": "Second Post", "body": "..."}
  ]
}
```

Advantages Of JSON Mock Databases

- Easy to set up and understand
- No additional dependencies required
- Can be version controlled with your codebase
- Useful for testing and prototyping

Limitations Of JSON Mock Databases

- Not suitable for concurrent writes (race conditions)
- Lack of advanced querying capabilities
- Entire file needs to be read/written for each operation
- Not scalable for large amounts of data or high traffic

Best Practices

- Use for prototyping and testing only
- Implement proper error handling
- Plan to migrate to a real database for production use

Topic

Express Router

What Is Express Router?

- A mini Express application
- Used for modular route handling
- Allows grouping of route handlers for a particular part of a site
- Helps in organizing routes across different files and modules

Why Use Express Router?

- Improves code organization
- Enhances maintainability
- Allows for middleware specific to a set of routes
- Facilitates versioning of API routes
- Makes the main app file cleaner and more modular

Best Practices For Using Router

- Group related routes together
- Use descriptive names for router files
- Keep routers focused on a specific feature or resource
- Use router parameters for shared logic
- Apply middleware at the router level when appropriate

Exercise

Poke Blog