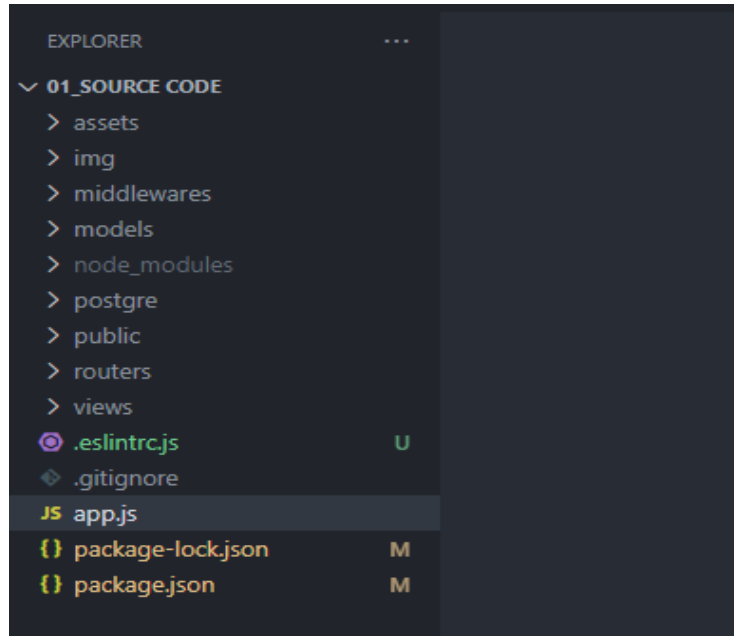


CODING STANDARDS

I. Project Structure Practices

1. Cấu trúc mã nguồn theo các thành phần một cách rõ ràng.

Phân chia mã nguồn thành các thành phần theo từng vai trò, mỗi thành phần có thư mục riêng và đảm bảo rằng mỗi thành phần được giữ ở mức nhỏ và đơn giản (API, routers, middlewares, models,...).



2. Đóng gói các thành phần common thành các packages

Trong một ứng dụng lớn tạo thành một cơ sở source code lớn, các tiện ích liên quan đến các quá trình như ghi log, mã hóa, v.v., nên được đóng gói và hiển thị dưới dạng npm packages.



3. Sử dụng các environment aware, secure and hierarchical config

Các biến môi trường và các biến cấu hình nên được đọc từ file config. File config nên có cấu trúc rõ ràng dễ đọc.

II. Error Handling Practices

1. Sử dụng Async-Await cho việc handle lỗi bất đồng bộ.

```
async function executeAsyncTask () {  
  
    try {  
  
        const valueA = await functionA();  
  
        const valueB = await functionB(valueA);  
  
        const valueC = await functionC(valueB);  
  
        return await functionD(valueC);  
  
    }  
  
    catch(err) {  
  
        logger.error(err);  
  
    }  
  
}
```

2. Sử dụng các Error object.

Nhiều lỗi được xử lý trả về dưới dạng chuỗi hoặc dạng mã một cách không thống nhất gây nhiều loạn, phức tạp việc xử lý lỗi và giảm khả năng tương tác giữa các module. Sử

dùng các error object làm tăng tính đồng nhất và tránh mất mát thông tin.

```
// centralized error object that derives from Node's Error

function AppError(name, httpCode, description, isOperational) {

    Error.call(this);

    Error.captureStackTrace(this);

    this.name = name;

    //...other properties assigned here

};
```

3. Phân biệt lỗi do vận hành và lỗi do lập trình viên.

- Operational errors (ví dụ như api nhận vào một tham số không hợp lệ) đề cập đến việc xác định lỗi có thể xảy ra và đã kiểm soát.
- Programmer error (truy cập biến mang giá trị null) đề cập tới việc lỗi chưa được kiểm soát.

```
// marking an error object as operational

const myError = new Error("How can I add new product when no value provided?");

myError.isOperational = true;

// or if you're using some centralized error factory

class AppError {

    constructor (commonType, description, isOperational) {

        Error.call(this);

        Error.captureStackTrace(this);

        this.commonType = commonType;

        this.description = description;

        this.isOperational = isOperational;

    }

};
```

4. **Sử dụng các trình kiểm soát lỗi API.**

Document API errors sử dụng Swagger hoặc GraphQL.

5. **Thoát tiến trình một cách có kiểm soát.**

Khi xảy ra lỗi không xác định – không chắc chắn về tình trạng hoạt động của ứng dụng. Thông lệ phổ biến đề xuất khởi động lại quy trình một cách cẩn thận bằng công cụ quản lý quy trình như Forever hoặc PM2.

6. **Test error flows sử dụng test framework.**

Dù là QA tự động chuyên nghiệp hay thử nghiệm dành cho nhà phát triển thủ công đơn giản – Đảm bảo rằng code chỉ đáp ứng các tình huống nghiệp vụ hiếm gặp mà còn xử lý và trả về đúng lỗi. Các test framework như Mocha & Chai có thể xử lý việc này một cách dễ dàng.

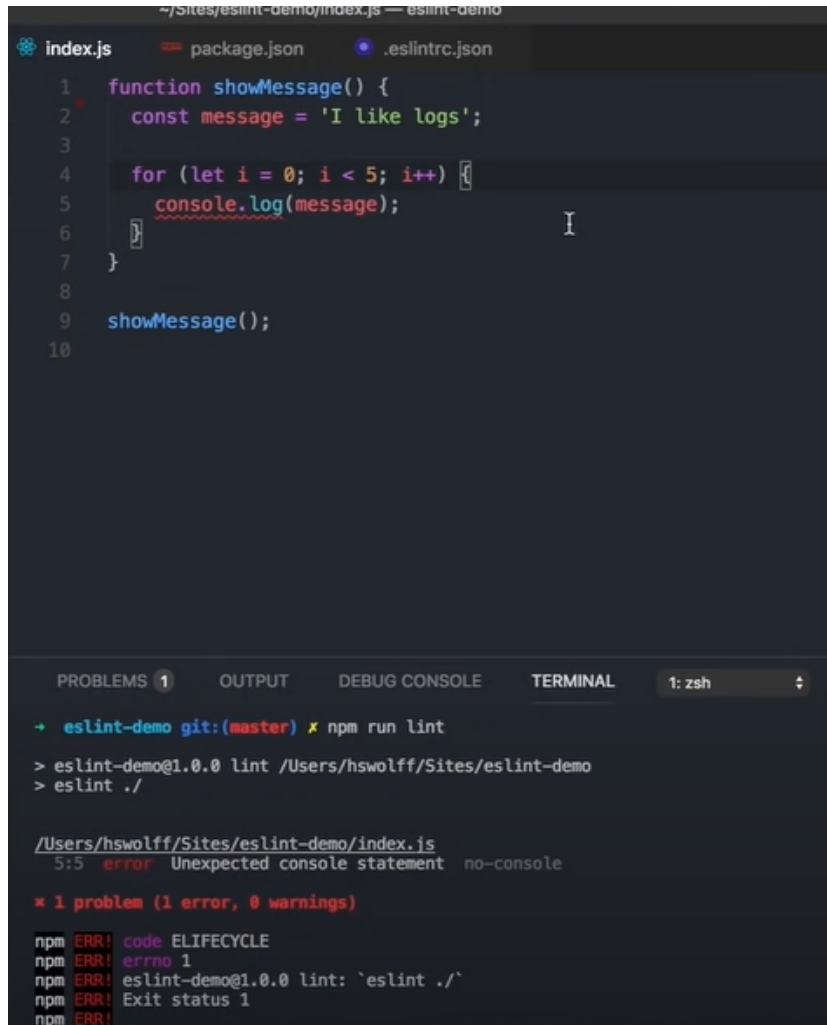
7. **Thực hiện validate bằng các thư viện chuyên dụng.**

Ví dụ như thư viện Joi hỗ trợ validate

III. Code Style Practices

ESLint là tiêu chuẩn thực tế để kiểm tra các lỗi code có thể xảy ra và fix, không chỉ để xác định các vấn đề về khoảng cách nghiêm trọng mà còn để phát hiện các lỗi mã

ng nghiêm trọng như các nhà phát triển đưa ra lỗi mà không cần phân loại.



The screenshot shows a VS Code editor with a file named `index.js`. The code contains a function `showMessage()` that logs a message. A linting error is highlighted on line 5: `Unexpected console statement`. The terminal at the bottom shows the command `npm run lint` being executed, which results in the same error message. The terminal also shows the output of the linting process, including the file path and the error details.

```
~/Sites/eslint-demo/index.js — eslint-demo
index.js package.json .eslintrc.json
1 function showMessage() {
2   const message = 'I like logs';
3
4   for (let i = 0; i < 5; i++) {
5     console.log(message);
6   }
7 }
8
9 showMessage();
10

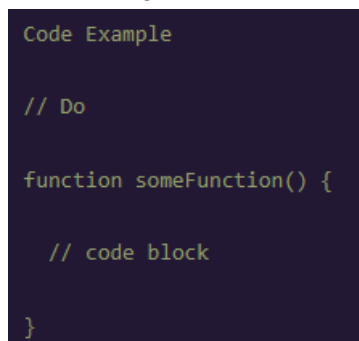
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL 1: zsh
+ eslint-demo git:(master) ✗ npm run lint
> eslint-demo@1.0.0 lint /Users/hswolff/Sites/eslint-demo
> eslint ./

/Users/hswolff/Sites/eslint-demo/index.js
5:5  error  Unexpected console statement  no-console

* 1 problem (1 error, 0 warnings)
npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! eslint-demo@1.0.0 lint: `eslint .`
npm ERR! Exit status 1
npm ERR!
```

+ Các qui tắc quan trọng:

- Dấu mở ngoặc của một khối code nên nằm trên cùng một dòng.



A code example snippet showing a function definition. The code is as follows:

```
Code Example

// Do

function someFunction() {

    // code block

}
```

- Phân tách các câu lệnh cho dễ nhìn.
- Đặt tên (biến, hàm, class,...) rõ ràng, dễ hiểu.
- Ưu tiên khai báo const thay vì let, var.
- Ưu tiên khai báo các module trước trong mỗi file.
- Sử dụng phép so sánh “===” thay vì “=”.
- Ưu tiên sử dụng Async await, tránh các callback.
- Ưu tiên sử dụng arrow function expression (=>).