CODING STANDARDS

Mục lục

I. Đặt tên biến rõ ràng:

- Sử dụng tên biến mô tả chức năng của nó.
- Tránh việc sử dụng tên biến ngắn gọn không rõ nghĩa.

Không tốt:

```javascript
let x = 5;
```


Tốt:

```javascript
let numberOfItems = 5;
```


- Tên biến nên viết bằng chữ thường.
- Tên hàm nên viết bằng chữ thường, bắt đầu bằng chữ cái viết hoa.
- Tên biến và hàm nên có độ dài vừa phải, dễ đọc và hiểu.

```javascript
// Biến
let numberOfItems = 5;
let userName = "John Doe";

// Hàm
function calculateTotalPrice() {
  // ...
}

function displayError(message) {
  // ...
}
```


II. Định dạng và thụt lề:

- Tuân theo một quy tắc định dạng cụ thể cho mã nguồn (ví dụ: tabs hoặc dấu cách).
- Sử dụng thụt lề một cách đồng nhất để làm cho mã nguồn dễ đọc.

Cài đặt trong VSCode:

- Cài đặt Prettier extension từ Visual Studio Code Marketplace.
- Thêm file .prettierrc vào dự án để cấu hình các quy tắc định dạng.

JSON

```
{
  "singleQuote": true,
  "semi": false,
  "tabWidth": 4,
  "printWidth": 80,
  "arrowParens": "avoid",
  "trailingComma": "none",
  "endOfLine": "auto",
  "htmlWhitespaceSensitivity": "strict",
  "overrides": [
    {
      "files": "*.json",
      "options": {
        "printWidth": 200
      }
    }
  ]
}
```

Vui lòng thận trọng khi sử dụng mã. Tìm hiểu thêm

III. Quản lý bộ nhớ và tài nguyên:

- Đảm bảo giải phóng tài nguyên một cách đúng đắn sau khi sử dụng xong.
- Tránh rò rỉ bộ nhớ và sử dụng cơ chế quản lý bộ nhớ một cách an toàn.

Không tốt:

JavaScript

```
let globalVariable = someObject;

function processData() {
 // Sử dụng globalVariable
 // ...
}
```

Vui lòng thận trọng khi sử dụng mã. Tìm hiểu thêm

Tốt:

JavaScript

```
function processData() {
 let localVariable = someObject;
 // Sử dụng localVariable
 // ...

 // Giải phóng tài nguyên sau khi sử dụng
 localVariable = null;
}
```

Vui lòng thận trọng khi sử dụng mã. Tìm hiểu thêm

IV. Xử lý ngoại lệ:

- Sử dụng cú pháp xử lý ngoại lệ một cách đồng nhất.
- Tránh sử dụng ngoại lệ để kiểm soát luồng điều khiển.

JavaScript

```
try {
 // Thử nghiệm một logic
} catch (err) {
 // Xử lý ngoại lệ
} finally {
 // Dòng mã cuối cùng không liên quan đến xử lý ngoại lệ
}
```

Vui lòng thận trọng khi sử dụng mã. Tìm hiểu thêm

V. Kiểm thử và Ghi chú:

- Viết mã kiểm thử đầy đủ để đảm bảo tính ổn định và độ tin cậy.
- Ghi chú rõ ràng và chi tiết để giải thích logic phức tạp hoặc phần mã quan trọng.

Không tốt:

```JavaScript
let variable = 10; // Set variable to 10
```


Tốt:

```JavaScript
let numberOfItems = 10; // Initialize the variable with the number of items
```


VI. Hiệu suất và Tối ưu hóa:

- Tránh việc tối ưu hóa trước khi cần thiết.
- Sử dụng công cụ đo hiệu suất để xác định và cải thiện điểm bottleneck.

```JavaScript
// Sử dụng công cụ đo hiệu suất để xác định và cải thiện điểm bottleneck
console.time('myFunction');
myFunction();
console.timeEnd('myFunction');
```


VII. Quy tắc về Quy mô Dự án:

- Áp dụng quy tắc SOLID và các nguyên tắc thiết kế khác để tạo ra mã linh hoạt và dễ bảo trì.
- Chia nhỏ mã nguồn thành các module nhỏ để dễ quản lý.

VIII. An toàn và Bảo mật:

- Kiểm tra và xử lý lỗ hổng bảo mật.
- Tránh sử dụng mã nguồn có thể gây nguy hiểm cho hệ thống.

IX. Tuân thủ Quy ước Ngôn ngữ:

Cài đặt trong VSCode:

- Cài đặt ESLint extension từ Visual Studio Code Marketplace.

- Tuân thủ các quy ước ngôn ngữ cụ thể thông qua cấu hình của ESLint.

X. Tài liệu hóa mã nguồn:

Cài đặt trong VSCode:

- Sử dụng các chức năng như Markdown Preview để xem trước tài liệu trong VSCode.
- Ghi tài liệu trực tiếp trong các file README.md hoặc tạo các file tài liệu riêng.