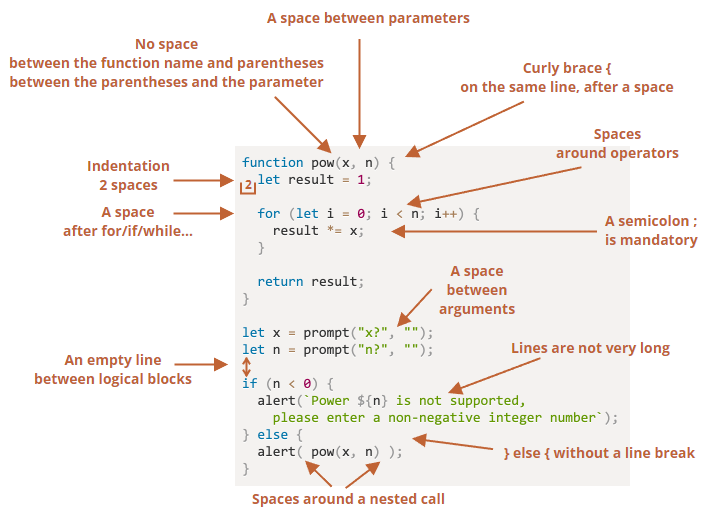
**Phần 2 Coding Style trong javascript**

Link code <https://github.com/dangtin306/baocao1>

**I . Giới thiệu**

+ Mã code rõ ràng và dễ đọc.

+ thực hiện một nhiệm vụ phức tạp và mã hóa nó theo cách vừa chính xác vừa có thể đọc được. Một kiểu mã tốt hỗ trợ rất nhiều trong việc đó.

**II . Cú pháp**

// nguồn https://javascript.info

Hình ảnh là một số quy tắc được đề xuất

**III Một số quy tắc viết mã**

### + [Dấu ngoặc nhọn](https://javascript.info/coding-style#curly-braces)

if (condition) {

// do this

// ...and that

// ...and that

}

Cấu trúc một dòng, chẳng hạn như if (condition) doSomething()

Dưới đây là các biến thể được chú thích để bạn có thể tự đánh giá khả năng đọc của chúng:

1. 😠 . Dấu ngoặc nhọn là không cần thiết:

if (n < 0) {alert(`Power ${n} is not supported`);}

1. 😠 Tách sang dòng riêng không dấu ngoặc. Dễ bị lỗi khi thêm dòng mới:

if (n < 0)

alert(`Power ${n} is not supported`);

1. 😏 Một dòng nếu ngắn có thể không có dấu ngoặc

if (n < 0) alert(`Power ${n} is not supported`);

1. 😃 Biến thể tốt nhất:

if (n < 0) {

alert(`Power ${n} is not supported`);

}

Đối với một mã rất ngắn gọn, một dòng được cho phép, ví dụ: if (cond) return null.

### + [Chiều dài dòng](https://javascript.info/coding-style#line-length)

Với một dòng mã dài. Cách tốt nhất là tách chúng ra.

Ví dụ:

// backtick quotes ` allow to split the string into multiple lines

let str = `

ECMA International's TC39 is a group of JavaScript developers,

implementers, academics, and more, collaborating with the community

to maintain and evolve the definition of JavaScript.

`;

Và, với điều kiện if :

if (

id === 123 &&

moonPhase === 'Waning Gibbous' &&

zodiacSign === 'Libra'

) {

letTheSorceryBegin();

}

Độ dài dòng tối đa phải thường là 80 hoặc 120 ký tự.

### + [thụt lề](https://javascript.info/coding-style#indents)

* **Thụt lề ngang: 2 hoặc 4 dấu cách.**

Thụt lề ngang được thực hiện bằng cách sử dụng 2 hoặc 4 dấu cách hoặc ký hiệu tab ngang (phím Tab).

Một lợi thế của khoảng trắng so với tab là khoảng trắng cho phép cấu hình thụt lề linh hoạt hơn so với biểu tượng tab.

Có thể căn chỉnh các tham số với dấu ngoặc mở, như thế này:

show(parameters,

aligned, // 5 spaces padding at the left

one,

after,

another

) {

// ...

}

* **Thụt lề dọc: các dòng trống để chia mã thành các khối logic.**

Ngay cả một chức năng đơn lẻ thường có thể được chia thành các khối hợp lý. Trong ví dụ bên dưới, việc khởi tạo các biến, vòng lặp chính và trả về kết quả được chia theo chiều dọc:

function pow(x, n) {

let result = 1;

// <--

for (let i = 0; i < n; i++) {

result \*= x;

}

// <--

return result;

}

Chèn thêm một dòng mới để giúp mã dễ đọc hơn. Không nên có nhiều hơn chín dòng mã mà không có thụt đầu dòng theo chiều dọc.

### [dấu chấm phẩy](https://javascript.info/coding-style" \l "semicolons)

Dấu chấm phẩy phải có sau mỗi câu lệnh .

Trong JavaScript, có những trường hợp ngắt dòng không được hiểu là dấu chấm phẩy, khiến mã dễ bị lỗi.

### Giảm thiểu xử lý

Ví dụ, trong vòng lặp, đôi khi nên sử dụng lệnh [continue](https://javascript.info/while-for#continue) để tránh lồng thêm.

Ví dụ, thay vì thêm một điều kiện lồng nhau if như thế này:

for (let i = 0; i < 10; i++) {

if (cond) {

... // <- one more nesting level

}

}

Có thể viết:

for (let i = 0; i < 10; i++) {

if (!cond) continue;

... // <- no extra nesting level

}

Một điều tương tự có thể được thực hiện với if/else và return.’

Ví dụ, hai cấu trúc bên dưới tương tự nhau nhau.

Lựa chọn 1:

function pow(x, n) {

if (n < 0) {

alert("Negative 'n' not supported");

} else {

let result = 1;

for (let i = 0; i < n; i++) {

result \*= x;

}

return result;

}

}

Lựa chọn 2:

function pow(x, n) {

if (n < 0) {

alert("Negative 'n' not supported");

return;

}

let result = 1;

for (let i = 0; i < n; i++) {

result \*= x;

}

return result;

}

Cái thứ hai dễ đọc hơn vì “trường hợp đặc biệt” của n < 0 được xử lý sớm. Sau khi kiểm tra xong, có thể chuyển sang luồng mã “chính” mà không cần lồng bổ sung.

## [Vị trí chức năng](https://javascript.info/coding-style" \l "function-placement)

Nếu đang viết một vài hàm “helper” và mã sử dụng, có ba cách để tổ chức các hàm.

1. Khai báo các chức năng trên mã sử dụng chúng:

// function declarations

function createElement() {

...

}

function setHandler(elem) {

...

}

function walkAround() {

...

}

// the code which uses them

let elem = createElement();

setHandler(elem);

walkAround();

1. Mã đầu tiên, sau đó chức năng

// the code which uses the functions

let elem = createElement();

setHandler(elem);

walkAround();

// --- helper functions ---

function createElement() {

...

}

function setHandler(elem) {

...

}

function walkAround() {

...

}

**IV Kiểm tra dòng mã và đưa ra các đề xuất cải thiện**

[ESLint](https://eslint.org/) là một công cụ phân tích tĩnh mã để nhanh chóng tìm ra sự cố. Nó được tích hợp vào hầu hết các trình soạn thảo mã và có thể chạy ESLint như một phần của quy trình tích hợp liên tục.

ESLint cung cấp extension cho các trình soạn mã khác nhau cài đặt extension của ESLint thì nó sẽ tự động thông báo lỗi realtime.

Cài đặt

Cài đặt ESLint thông qua npm ở dạng global.

$ npm i -g eslint

Sau đó chạy init để khởi tạo cấu hình ESLint cho project.

$ eslint --init

Khi chạy --init, ESLint sẽ hỏi một số câu hỏi để tạo cấu hình phù hợp, trả lời theo đúng những gì mà dự án đang sử dụng. Sau đó một file .eslintrc.js sẽ được tạo ra có chứa các cấu hình mà ESLint sẽ sử dụng cho dự án . (Lưu ý nếu bạn chọn cấu hình dưới dạng file json hoặc yaml thì ESLint sẽ tạo file cấu hình tương ứng).

Một cấu hình mặc định:

module.exports = {

"env": {

"browser": true,

"es2021": true

},

"extends": "eslint:recommended",

"overrides": [

],

"parserOptions": {

"ecmaVersion": "latest",

"sourceType": "module"

},

"rules": {

}

}

Cú pháp cơ bản để bắt đầu chạy Lint là:

$ eslint [options] [file|dir|glob]\*

Ví dụ chạy Lint cho file index.js

$ eslint index.js

ESLint sẽ tự động tìm tệp cấu hình .eslintrc trong dự án và Lint theo cấu hình đó .

Giả sử tất cả mã nằm trong thư mục /src và chạy Lint toàn bộ dự án:

$ eslint src

Với các tệp .js và hãy bỏ qua các file trong thư mục src/dist:

$ eslint src --ignore-pattern 'src/dist' --ext .js

Còn nhiều lệnh CLI khác nữa có thể tham khảo tại [trang tài liệu ESLint](https://eslint.org/docs/latest/user-guide/command-line-interface).

ESLint còn có thể tự động sửa lỗi vi phạm các rule bằng cách chạy --fix:

# Sửa lỗi nhưng không lưu lại

$ eslint src --fix-dry-run

# Sửa lỗi và lưu lại

$ eslint src --fix