Java script - AngularJS Style Guide

VCS - PRODUCT DEVELOPMENT STANDARDS

Exported on Sep 18, 2020

Table of Contents

1 Mục lục 4

2 Giới thiệu 5

3 Guidelines 6

3.1 Quy tắc đặt tên 6

3.1.1 Quy tắc chung 6

3.1.2 Tên package 6

3.1.3 Tên class 6

3.1.4 Tên method 6

3.1.5 Tên hằng số 6

3.1.6 Tên trường, tên tham số, tên biến 6

3.2 Comment 7

3.3 Sử dụng dấu chấm phẩy 7

4 Style 8

4.1 File 8

4.2 Format 8

4.2.1 Cặp ngoặc nhọn {} 8

4.2.2 Block indentation 9

4.2.3 Mảng - Array 9

4.2.4 Dict Object 10

4.2.5 Switch case 10

4.2.6 Column limit 11

4.2.7 Line-wrapping 11

4.2.8 Sử dụng khoảng trắng (whitespace) 12

4.2.9 Sử dụng dòng trống (blank line) 12

Tài liệu này được tổng hợp dựa trên các nguồn sau:

* [Google JavaScript Style Guide](https://google.github.io/styleguide/jsguide.html#introduction)
* [AngularJS Style Guide](https://google.github.io/styleguide/angularjs-google-style.html)

# Mục lục

* [Giới thiệu](#scroll-bookmark-3)
* [Guidelines](#scroll-bookmark-4)
  + [Quy tắc đặt tên](#scroll-bookmark-5)
    - [Quy tắc chung](#scroll-bookmark-6)
    - [Tên package](#scroll-bookmark-7)
    - [Tên class](#scroll-bookmark-8)
    - [Tên method](#scroll-bookmark-9)
    - [Tên hằng số](#scroll-bookmark-10)
    - [Tên trường, tên tham số, tên biến](#scroll-bookmark-11)
  + [Comment](#scroll-bookmark-12)
  + [Sử dụng dấu chấm phẩy](#scroll-bookmark-13)
* [Style](#scroll-bookmark-14)
  + [File](#scroll-bookmark-15)
  + [Format](#scroll-bookmark-16)
    - [Cặp ngoặc nhọn {}](#scroll-bookmark-17)
    - [Block indentation](#scroll-bookmark-18)
    - [Mảng - Array](#scroll-bookmark-19)
    - [Dict Object](#scroll-bookmark-20)
    - [Switch case](#scroll-bookmark-21)
    - [Column limit](#scroll-bookmark-22)
    - [Line-wrapping](#scroll-bookmark-23)
    - [Sử dụng khoảng trắng (whitespace)](#scroll-bookmark-24)
    - [Sử dụng dòng trống (blank line)](#scroll-bookmark-25)

# Giới thiệu

Tài liệu hướng dẫn này hướng dẫn chi tiết về các điều **Nên** hoặc **Không nên** khi lập trình frontend sử dụng Java script hay Angular JS.

# Guidelines

## Quy tắc đặt tên

### Quy tắc chung

* Sử dụng chữ cái và số, không dùng ký tự đặc biệt. Hạn chế dùng dấu gạch dưới ( \_ ), trừ 1 số trường hợp đặc biệt được sử dụng bởi các thư viện (lib).
* Đặt tên có ý nghĩa để khi người dùng nhìn tên hàm, biến... có thể hiểu ngay được ý nghĩa của nó. Không sử dụng các từ viết tắt mơ hồ hoặc xa lạ với người đọc.

|  |  |
| --- | --- |
| **Bad** | **Good** |
| |  | | --- | | abc // Không có ý nghĩa  nErr // Từ viết tắt mơ hồ  nCompConns // Từ viết tắt mơ hồ  pcReader // Quá nhiều từ có thể viết tắt là "pc"  cstmrId // Người khác không hiểu được "cstmr" là gì | | |  | | --- | | errorCount // Ý nghĩa rõ ràng, không viết tắt  dnsConnectionIndex // Hầu hết mọi người đều hiểu "dns" là gì. Đây là từ viết tắt được sử dụng phổ biến trên thế giới.  referrerUrl // Tương tự như trên  customerId // Chỉ trường định danh thông tin của khách hàng, không có khả năng gây hiểu nhầm | |

### Tên package

* Sử dụng dạng lowerCamelCase, không có ký tự gạch dưới. Ví dụ: my.exampleCode.deepSpace, không sử dụng my.examplecode.deepspace hoặc  my.example\_code.deep\_space.

### Tên class

* Sử dụng dạng UpperCamelCase, thường sử dụng danh từ hoặc cụm danh từ. Ví dụ: Group, NewCustomer,...
* Class test luôn có hậu tố là Test. Ví dụ: HashTest

### Tên method

* Sử dụng dạng lowerCamelCase
* Tên method luôn là động từ hoặc cụm động từ. Ví dụ sendMessage hoặc stop

### Tên hằng số

* Sử dụng dạng UPPER\_CASE\_WITH\_UNDERSCORES (tất cả là chữ hoa, tách nhau bởi gạch dưới)

### Tên trường, tên tham số, tên biến

* Sử dụng dạng lowerCamelCase

## Comment

* Block comment

|  |
| --- |
| // Comment a line  /\* Comment a paragraph \*/  /\*  \* Comment a long multi-line paragraph  \* This  \* is  \* multiple  \* line  \*/ |

* Parameter comment: nên được sử dụng khi giá trị và tên phương thức không truyền đạt đủ ý nghĩa. Có thể đặt comment trước hoặc sau giá trị truyền vào tham số trong phương thức.

|  |
| --- |
| // Set parameter comment before each parameter value  someFunction(obviousParam, /\* shouldRender= \*/ **true**, /\* name= \*/ 'hello');      // Set parameter comment after each parameter value  someFunction(obviousParam, **true** /\* shouldRender \*/, 'hello' /\* name \*/); |

## Sử dụng dấu chấm phẩy

Luôn sử dụng dấu chấm phẩy ( ; ) khi kết thúc một câu lệnh.

|  |
| --- |
| // Variables  **var** values = ["Volvo", "Saab", "Fiat"];    **var** person = {  firstName: "John",  lastName: "Doe",  age: 50,  eyeColor: "blue"  };    // Function  **function** toCelsius(fahrenheit) {  **return** (5 / 9) \* (fahrenheit - 32);  }    // Loop  **for** (i = 0; i < 5; i++) {  x += i;  }    // Conditional  **if** (time < 20) {  greeting = "Good day";  } **else** {  greeting = "Good evening";  } |

# Style

## File

* Tên file phải được đặt bằng chữ thường, không dấu, có thể bao gồm dấu gạch ngang ( - ) hoặc dấu gạch dưới ( \_ ). Phần mở rộng của tên file phải là .js .
* Source file luôn được encode là UTF-8

## Format

### Cặp ngoặc nhọn {}

* Cặp ngoặc nhọn {} được sử dụng trong các lệnh if, else, for, do và while, chứa khối lệnh thực thi, ngay cả khi khối lệnh này rỗng hoặc chỉ chứa một lệnh duy nhất.
* Câu lệnh đầu tiên của một khối trong cặp ngoặc nhọn {} phải bắt đầu trên dòng riêng của nó.

|  |  |
| --- | --- |
| **Bad** | **Good** |
| |  | | --- | | **if** (someVeryLongCondition())  doSomething();    **for** (let i = 0; i < foo.length; i++) bar(foo[i]); | | |  | | --- | | **if** (someVeryLongCondition()) {  doSomething();  }    **for** (let i = 0; i < foo.length; i++) {  bar(foo[i]);  } | |

* Quy tắc sử dụng ngoặc nhọn theo K & R style (Kernighan and Ritchie style) cho khối lệnh không rỗng như sau:
  + Dấu mở ngoặc { không phải là ký tự đầu tiên khác rỗng của dòng (do đó mở ngoặc luôn nằm cùng dòng với lệnh if, else, for, do và while)
  + Sau kí tự mở ngoặc { phải xuống dòng
  + Luôn xuống dòng trước khi sử dụng dấu đóng ngoặc } hay nói cách khác dấu đóng ngoặc } luôn là ký tự đầu tiên của dòng
  + Sau dấu đóng ngoặc } chỉ xuống dòng chỉ khi nó là kết thúc hoàn toàn câu lệnh hoặc kết thúc method, class (ví dụ với dấu đóng ngoặc } của lệnh if. Trong trường hợp có sử dụng else, catch, while thì sẽ không xuống dòng)

Ví dụ:

|  |
| --- |
| class InnerClass {  constructor() {}    /\*\* @param {number} foo \*/  method(foo) {  **if** (condition(foo)) {  **try** {  // Note: this might fail.  doSomething();  } **catch** (err) {  recover();  }  }  }  } |

* Nếu khối lệnh trong cặp ngoặc nhọn là rỗng thì có thể viết đơn giản trên cùng dòng ***{}.***Ngoại trừ khối lệnh rỗng này là một phần của khối multi-block khác như if/else, try/catch/finally

|  |  |
| --- | --- |
| **Bad** | **Good** |
| |  | | --- | | **if** (condition) {  // ...  } **else** **if** (otherCondition) {} **else** {  // ...  }    **try** {  // ...  } **catch** (e) {} | | |  | | --- | | **function** doNothing() {}    **if** (condition) {  // ...  } **else** **if** (otherCondition) {  } **else** {  // ...  }    **try** {  // ...  } **catch** (e) {  } | |

### Block indentation

* Sử dụng 2 khoảng trắng để indentation các khối (trước đây thường dùng 4 khoảng trắng nhưng ngày nay nên dùng 2 khoảng trắng để tương thích với chuẩn của GitHub). Khi tạo một khối mới phải indentation 2 khoảng trắng, sau khi kết thúc khối lại thụt ra mức indentation trước đó
* Không sử dụng tab để indentation

### Mảng - Array

* Một mảng có thể được khai báo với format tùy ý như các ví dụ bên dưới. Tuy nhiên nên khai báo các phần tử trong mảng trên 1 dòng để giảm kích thước về chiều dọc (vertical size), đặc biệt là đối với các mảng có số lượng phần tử lớn.

|  |
| --- |
| const a = [  0,  1,  2,  ];    const b =  [0, 1, 2];    // Recommend  const c = [0, 1, 2];    someMethod(foo, [  0, 1, 2,  ], bar); |

### Dict Object

* Khai báo tương tự như mảng, tất cả các khai báo bên dưới đều hợp lệ.

|  |
| --- |
| const a = {  a: 0,  b: 1,  };    const b =  {a: 0, b: 1};    const c = {a: 0, b: 1};    someMethod(foo, {  a: 0, b: 1,  }, bar); |

### Switch case

* Sử dụng quy tắc thụt lề 2 khoảng trắng
* Sau mỗi câu lệnh break, sử dụng 1 line trống đề tách với case tiếp theo

|  |  |
| --- | --- |
| **Bad** | **Good** |
| |  | | --- | | **switch** (animal) {  **case** Animal.BANDERSNATCH:  handleBandersnatch();  **break**;  **case** Animal.JABBERWOCK:  handleJabberwock();  **break**;  **default**:  **throw** **new** Error('Unknown animal');  } | | |  | | --- | | **switch** (animal) {  **case** Animal.BANDERSNATCH:  handleBandersnatch();  **break**;    **case** Animal.JABBERWOCK:  handleJabberwock();  **break**;    **default**:  **throw** **new** Error('Unknown animal');  } | |

### Column limit

* JavaScript code giới hạn 80 ký tự (cột) trên 1 dòng, trừ các trường hợp sau:
  + goog.module, goog.require, goog.requireType statements
  + ES module import and export from statements
  + Các dòng không thể tuân theo giới hạn cột hoặc việc tuân theo giới hạn này sẽ cản trở khả năng đọc, ví dụ như:
    - Một URL dài cần phải click vào để đến trang đích tương ứng
    - Một shell command hướng đến việc hỗ trợ copy và paste
    - Một file path dài

### Line-wrapping

Thuật ngữ line-wrapping chỉ việc break một đoạn mã thành nhiều dòng để tuân thủ quy tắc giới hạn cột được mô tả ở trên. Nguyên tắc thực hiện như sau:

* Xuống dòng sau một dấu phẩy

|  |
| --- |
| let result = someMethod(longExpression1, longExpression2, longExpression3,  longExpression4, longExpression5);    let result = someMethod1(longExpression1,  someMethod2(longExpression2,  longExpression3)); |

* Xuống dòng trước một toán tử (ví dụ +, -)

|  |
| --- |
| longName1 = longName2 \* (longName3 + longName4 - longName5)  + 4 \* longname6; |

* Không có công thức toàn diện, xác định xem nên break ở đoạn nào trong mọi tình huống là hợp lý nhất, tuy nhiên nên ưu tiên xuống dòng lần lượt từ mức cao (higher-level) rồi mới đến mức thấp (lower-level)

|  |  |
| --- | --- |
| **Bad** | **Good** |
| |  | | --- | | currentEstimate = calc(currentEstimate + x \*  currentEstimate) / 2.0; | | |  | | --- | | currentEstimate =  calc(currentEstimate + x \* currentEstimate) /  2.0; | |

* Sử dụng 4 khoảng trắng để indentation (thụt dòng) giữ các dòng

### Sử dụng khoảng trắng (whitespace)

Sử dụng một khoảng trắng trong một số trường hợp sau:

* Tách các keyword **if**, **for**, **catch** với dấu mở ngoặc ( đằng sau nó. Ví dụ ***if (a == 1)***
* Tách các keyword **else**, **catch** với ký tự đóng ngoặc nhọn } đằng trước nó. Ví dụ ***} else {***
* Trước bất kỳ dấu mở ngoặc nhọn nào {. Ngoại trừ 2 trường hợp:
  + Trước một object là tham số đầu tiên của function  hoặc phần tử đầu tiên trong mảng (ví dụ: foo({a: [{c: d}]}))
  + Trong một template expansion, ví dụ: biểu thức hợp lê: x = `ab${1 + 2}cd`; không hợp lệ: x = `ab$ {1 + 2}z`)
* Trước và sau của bất kỳ toán tử nhị phân nào ví dụ: ***a && b***
* Sau dấu phẩy ( , ) hoặc dấu chấm phẩy ( ; ). Chú ý không được sử dụng khoảng trắng trước 2 dấu này.
* Sau dấu hai chấm ( : ) khi khai báo một object dạng dict
* Trước và sau double slash ( // )
* Sau các parameter comment: baz(/\* buzz= \*/ true)

### Sử dụng dòng trống (blank line)

Sử dụng một dòng trống để phân cách giữa các khối lệnh hay logic không liên quan đến nhau.

|  |  |
| --- | --- |
| **Bad** | **Good** |
| |  | | --- | | **function** doSomething() {  // do something  }  **function** doSomethingElse() {  // do something else  } | | |  | | --- | | **function** doSomething() {  // do something  }    **function** doSomethingElse() {  // do something else  } | |