# 重庆紫光华山智安科技有限公司技术规范

# 安防产品前端编码规范

目录

# 1. 简介

本前端代码规范以 W3C 标准为基础，结合 GitHub 代码库中业界优秀编码以及我司实际应用场景进行调整。

# 2. Naming Project

## 2.1. 项目命名

项目名全部采用小写方式， 以中划线分隔。 项目命名遵循：语言+项目+类型的命名方式。比如：react-video-ststem-web。

## 2.2. 目录名称

目录名也采用小写方式， 以中划线分隔。有复数结构时，要采用复数命名法，比如说： scripts, styles, images

## 2.3. HTML 文件命名

HTML 参照上一条规则，目录中主文件统一使用 index 命名，其他文件根据功能命名。

## 2.4. Css 文件命名

Css 参照上一条规则,并需与相对应 HTML 同名。

## 2.5. JavaScript 文件命名

JavaScript 参照上一条规则 ,但 JavaScript 应对类文件进行首字母大写或所在目录进行首字母大写。

# 3. HTML

## 3.1. 语法

- 使用四个空格的 soft tabs — 这是保证代码在各种环境下显示一致的唯一方式。
- 嵌套的节点应该缩进（四个空格）。
- 在属性上，使用双引号，不要使用单引号。
- 不要在自动闭合标签结尾处使用斜线 - HTML5 规范 指出他们是可选的。
- 不要忽略可选的关闭标签。

## 3.2. HTML5 doctype

在每个 HTML 页面开头使用这个简单地 doctype 来启用标准模式，使其每个浏览器中尽可能一致的展现。统一使用 HTML5 文档类型： <!DOCTYPE html>。

## 3.3. Language attribute

声明当前页面的语言类型。中文对应 ZH，英文对应 EN。规定使用<html lang="zh-cmn-Hans"></html>

## 3.4. IE compatibility mode

设置 <meta http-equiv="X-UA-Compatible" content="IE=Edge"> ，IE8/9 及以后的版本都会以最高版本 IE 来渲染页面。

## 3.5. 字符编码

通过声明一个明确的字符编码，让浏览器轻松、快速的确定适合网页内容的渲染方式。规定使用：<meta charset="UTF-8">。

## 3.6. TDK

### 3.6.1. 页面标题

应用名称-产品名称。不要超过 25 个中文。比如：<title>图上监控-两江公安视综平台</title>

### 3.6.2. 页面关键字

产品，应用有关名词，之间用英文半角逗号隔开。比如：<meta name="keywords" content="视频,监控" />

### 3.6.3. 页面描述

不超过 150 个字符，描述内容要和页面内容相关。比如：<meta name="description" content="内容"/>

## 3.7. 移动端 Meta

设置视口地宽度为设备宽度，如需对 retina 屏做处理，自行根据 dpr 计算。规定使用：

<meta name="viewport" content="width=device-width,initial-scale=1,mi

nimum-scale=1,maximu m-scale=1,user-scalable=no">。

## 3.8. 引入 Css 和 JavaScript

根据 HTML5 规范，通常在引入 CSS 和 JavaScript 时不需要指明 type，因为 text/css 和 text/javascript 分别是他们的默认值。

## 3.9. 实用高于完美

尽量遵循 HTML 标准和语义，但是不应该以浪费实用性作为代价。任何时候都要用尽量小的复杂度和尽量少的标签来解决问题。比如:table

## 3.10. 减少标签数量

在编写 HTML 代码时，需要尽量避免多余的父节点。很多时候，需要通过迭代和重构来使 HTML 变得更少。

## 3.11. 代码分离

严格地保证 HTML、CSS、JS 三者分离，并尽量使三者之间没有太多的交互和联系。

- 尽可能减少 CSS 和 JS 引用个数，尽量合并压缩
- 不使用行内样式。
- 不在元素上使用 style 属性。
- 不使用行内脚本。

### 3.12. JavaScript 生成标签

在 JavaScript 文件中生成标签让内容变得更难查找，更难编辑，性能更差。应该尽量避免这种情况的出现。如真需要使用，请使用第三方 web 模板。

# 4. CSS

## 4.1. 语法

- 使用四个空格的 soft tabs。
- 使用组合选择器时，保持每个独立的选择器占用一行。
- 为了代码的易读性，在每个声明的左括号前增加一个空格。
- 声明块的右括号应该另起一行。
- 每条声明后应该插入一个空格。
- 每条声明应该只占用一行来保证错误报告更加准确。
- 所有声明应该以分号结尾。
- 所有的十六进制值都应该使用小写字母，例如 #fff。
- 逗号分隔的取值，都应该在逗号之后增加一个空格。比如说 box-shadow。
- 不要为 0 指明单位。

```
.selector,
.selector-secondary,
.selector[type="text"] {
    padding: 15px;
    margin-bottom: 15px;
    background-color: rgba(0,0,0,.5);
```

```
    box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

## 4.2. Class 命名

- 保持 Class 命名为全小写，可以使用短划线。
- class 用于标识某一个类型的对象，命名必须言简意赅。
- 规则名称中不应该包含颜色（red/blue）、定位（left/right）等与具体显示效果相关的信息。

  应该用功能命名，而不是样式显示结果命名。
- 使用 .j-* classes 来表示行为。

## 4.3. Don't use @import

@import 速度较慢，会增加额外的页面请求，并可能导致其他无法预见的问题。

## 4.4. 媒体查询位置

尽量将媒体查询的位置靠近他们相关的规则。不要将他们一起放到一个独立的样式文件中 ,或者丢在文档的最底部。这样做只会让大家以后更容易忘记他们。

例如：

```
.element { ... }
.element-avatar { ... }
.element-selected { ... }
@media (min-width: 480px) {
    .element { ...}
    .element-avatar { ... }
    .element-selected { ... }
}
```

## 4.5. 前缀属性

当使用厂商前缀属性时，通过缩进使取值垂直对齐以便多行编辑。推荐使用 autoprefixer 自动补全插件。

## 4.6. 属性简写

坚持限制属性取值简写的使用，属性简写需要你必须显式设置所有取值。

## 4.7. Less 和 Sass 中的嵌套

避免不必要的嵌套。可以进行嵌套，不意味着你应该这样做。只有在需要给父元素增加样式并且同时存在多个子元素时才需要考虑嵌套。

## 4.8. 代码组织

- 以组件为单位组织代码。
- 制定一个一致的注释层级结构。
- 使用一致的空白来分割代码块，这样做在查看大的文档时更有优势。
- 当使用多个 CSS 文件时，通过组件而不是页面来区分他们。页面会被重新排列，而组件移动就可以了

## 4.9. 代码注释

保证你的代码是描述性的，包含好的注释，并且容易被他人理解。好的代码注释传达上下文和目标。不要简单地重申组件或者 class 名称。

```
/* Bad example */
/* Modal header */
```

```
.modal-header {
    ...
}

/* Good example */
/* Wrapping element for .modal-title and .modal-close
*/.modal-header {
    ...
}
```

# 5. JavaScript

## 5.1. 语法

- 一律使用 4 个空格
- 继续缩进同样适用 4 个空格，跟上一行对齐。
- 声明之后一律以分号结束，不可以省略。
- 单行长度，理论上不要超过 80 列。
- 如果需要换行，存在操作符的情况，一定在操作符后换行，然后换的行缩进 4 个空格。
- 统一使用双引号。

```
if (typeof qqfind === "undefined" ||
    typeof qqfind.cdnrejected === "undefined" ||
    qqfind.cdnrejected !== true) {
    url = "http://pub.idqqimg.com/qqfind/js/location4.js";
} else {
    url = "http://find.qq.com/js/location4.js";
}
```

## 5.2. 空行

- 方法之间加。
- 单行或多行注释前加。
- 逻辑块之间加空行增加可读性。

## 5.3. 变量命名

- 标准变量采用驼峰标识。
- 使用的 ID 的地方一定全大写。
- 使用的 URL 的地方一定全大写，比如说 reportURL。
- 涉及 Android 的，一律大写第一个字母。
- 涉及 iOS 的，一律小写第一个，大写后两个字母。
- 常量采用大写字母，下划线连接的方式。

```
var thisIsMyName;

var goodID;

var AndroidVersion;

var iOSVersion;

var MAX_COUNT = 10;
```

## 5.4. 方法和类命名

- 构造函数和类名词首字母大写。
- 方法命名一般常用动宾结构。
- 查询方法要查询的内容在前，条件在后。

- 类中私有方法和属性采用下划线方式。

```
function Person(name) {
    this.name = name
}

class Person {
    _name = xxx
}
```

## 5.5. undefined 使用场景

- 永远不要直接使用 undefined 进行变量判断。
- 使用字符串 "undefined" 对变量进行判断。

## 5.6. 单行注释

- 双斜线后，必须跟注释内容保留一个空格。
- 可独占一行，前边不许有空行，缩进与下一行代码保持一致。
- 可位于一个代码行的末尾，注意这里的格式。

```
if (condition) {

    // if you made it here, then all security checks passed
    allowed();
}

var zhangsan = "zhangsan";    // 双斜线距离分号四个空格，双斜线后始终保留一个空格
```

## 5.7. 多行注释格式

- 最少三行

- 前边留空一行
- 对于难于理解的代码段 ,可能存在错误的代码段 ,浏览器特殊的 HACK 代码，业务逻辑强相关的代码使用。

```
/*
 *  注释内容与星标前保留一个空格
 */
```

## 5.8. 文档注释

- 各类标签 @param @method 等 参考 http://usejsdoc.org/
- 所有的函数，构造函数，对象类方法使用。

## 5.9. if else else 前后留有空格

```
if (condition) {
    doSomething();
} else {
    doSomethingElse();
}
```

## 5.10. for

- 普通for循环，分号后留有一个空格，判断条件等内的操作符两边不留空格，前置条件如果有多个，逗号后留一个空格。
- for-in 一定要有 hasOwnProperty 的判断， 否则 JSLint 或者 JSHint 都会有一个 warn。

## 5.11. 变量声明

- 所有函数内变量声明放在函数内头部，只使用一个 var。

- 全局使用 var。
- 函数或者块作用域中使用 let，常量使用 const。

### 5.12. 杂项

- 完全避免 == != 的使用， 用严格比较条件 === !== 。
- eval 非特殊业务， 禁用。
- with 非特殊业务， 禁用。

# 6. 编辑器配置 & 检测工具

## 6.1. 编辑器配置

不同编辑器的默认格式不同，使用 editorConfig 可以帮助开发人员在不同的编辑器和 IDE 中定义和维护一致的编码风格。配置方法：https://editorconfig.org/

- 使用四个空格的 soft-tabs。
- 在保存时删除尾部的空白字符。
- 设置文件编码为 UTF-8。
- 在文件结尾添加一个空白行。为什么这么做？因为 在 Unix 中 \n 符号被定义为一行的『结束符』，如果一行的结尾没有 \n 视为这一行没有结束，换句话说这个文件不完整，也就是说这根本不是一个合法的文本文件。

## 6.2. 检测工具 ESLint

常见的检测工具有 jsLint，jsHint，ESLint 。为什么我们使用 esLint。因为 esLint 相对于前面两者配置更加灵活，易于迁移，而且 esLint 是唯一支持 jsx 的工具。所以我们选择使用它。

### 6.2.1. ESLint 主要配置

- Parse： ESLint 默认使用 Espree 作为其解析器，但是因为我们使用了 ES Lint 默认解析器无法识别地语法，所以我们使用 Babel-ESLint 代替默认解析器。
- Env: 一个环境定义了一组预定义的全局变量。指定你想启用的环境，并设置它们为 true。
- Extends：一个配置文件可以被基础配置中的已启用的规则继承。我们使用社区流行的代码风格 airbnb，prettier 和自定义规则相结合。
- Rules: 自定义规则。

### 6.2.2. Airbnb

AirBnb 是社区比较热门的代码规范。其规则众多，所以本文档只介绍部分常用的规则。

#### (1) 引用

- 对所有的引用使用 const ，不要使用 var。
- 可变动的引用，使用 let 代替 var。

### (2) 对象

- 使用字面值创建对象。
- 使用对象方法的简写。
- 使用对象属性值的简写。
- 在对象属性声明前把简写的属性分组。
- 不要直接调用 Object.prototype 的方法。
- 浅拷贝对象的时候最好是使用扩展符

### (3) 数组

- 使用字面值创建数组。
- 使用扩展符复制数值。
- 使用 Array#from 把一个类数组对象转换成数组。

### (4) 函数

- 使用函数声明代替函数表达式。
- 永远不要在一个非函数代码块（if、while 等）中声明一个函数，把那个函数赋给一个变量。浏览器允许你这么做，但它们的解析表现不一致。
- 不要使用 arguments。
- 如果一个函数适合用一行写出并且只有一个参数，那就把花括号、圆括号和 return 都省略掉。

### (5) 构造器

- 总是使用 class，而不是构造函数。

- 使用 extends 继承，而不是自定义原型继承。

#### (6) 模块

- 使用 import/export，而不是 require。
- 不要使用通配符 import。
- 不要从 import 中直接 export。

```
// bad
export { es6 as default } from './airbnbStyleGuide';

// good
import { es6 } from './AirbnbStyleGuide';
export default es6;
```

#### (7) 比较运算符 & 等号

- 优先使用 === 和 !== 而不是 == 和 !=.

### 6.2.3. Prettier

Why Prettier?使用 Airbnb 对代码进行规范，但它重点却是检测代码中潜在的问题，并没法解决代码风格的问题。而 Prettier 可以很好地弥补这个问题。Prettier 可以单独使用，根据我们实际情况将 Prettier 配合 ESLint 使用。

一、ESLint 与 Prettier 配合使用方法

- 首先安装 prettier,eslint-plugin-prettier,eslint-config-prettier 三个模块。
- 然后配置.eslintrc 文件 extends 属性为：["plugin:prettier/recommended"]。

二、对 Prettier 进行配置

我们将通过配置.prettierrc 文件的方式，执行之前制定的代码风格规范。配置如下：

```
"printWidth": 80, //一行的字符数，如果超过会进行换行，默认为 80
"tabWidth": 4, //一个 tab 代表几个空格数，默认为 80
"useTabs": false, //是否使用 tab 进行缩进，默认为 false，表示用空格进行缩减
"singleQuote": false, //字符串是否使用单引号，默认为 false，使用双引号
"semi": true, //行位是否使用分号，默认为 true
"trailingComma": "es5", //是否使用尾逗号，有三个可选值"<none|es5|all>"
"bracketSpacing": true, //对象大括号直接是否有空格，默认为 true，
```

除了配置以上属性外，还有：

- parser 指定解析引擎，因为我们使用 babel-eslint 引擎，所以就不再配置。
- Overrides 指定特殊文件进行单独配置。现在暂无需要。
- alwaysParens 设置为单行箭头函数的参数添加圆括号，设置默认不添加圆括号。
- FilePath 指定文件的输入路径，这将被用于解析器参照。单独使用 Prettier 时设置。