前端项目开发规范（腾讯alloyteam团队）

最佳原则

坚持制定好的代码规范。
无论团队人数多少，代码应该同出一门。

命名规范

- **项目命名**
    - 全部采用小写方式， 以下划线分隔。
        - 例：my_project_name
- **目录命名**
    - 参照项目命名规则；
    - 有复数结构时，要采用复数命名法。
        - 例：scripts, styles, images, data_models
- **JS文件命名**
    - 参照项目命名规则；
        - 例：account_model.js
- **CSS, SCSS文件命名**
    - 参照项目命名规则。
        - 例：retina_sprites.scss
- **HTML文件命名**
    - 参照项目命名规则。
        - 例：error_report.html

HTML

- **语法**
    - 缩进使用soft tab（4个空格）；
    - 嵌套的节点应该缩进；
    - 在属性上，使用双引号，不要使用单引号；

- 属性名全小写，用中划线做分隔符；
- 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，例： 和 </body>。

```
<!DOCTYPE html>

<html>

  <head>

    <title>Page title</title>

  </head>

  <body>

    <img src="images/company_logo.png" alt="Company">

    <h1 class="hello-world">Hello, world!</h1>

  </body>

</html>
```

- **HTML5 doctype**
  - 在页面开头使用这个简单地doctype来启用标准模式，使其在每个浏览器中尽可能一致的展现；
  - 虽然doctype不区分大小写，但是按照惯例，doctype大写 （关于html属性，大写还是小写）。

```
<!DOCTYPE html>

<html>

  ...

</html>
```

- **lang属性**
  - 根据HTML5规范：
  - 应在html标签上加上lang属性。这会给语音工具和翻译工具帮助，告诉它们应当怎么去发音和翻译。
  - 更多关于 lang 属性的说明在这里；
  - 在sitepoint上可以查到语言列表；
  - 但sitepoint只是给出了语言的大类，例如中文只给出了zh，但是没有区分香港，台湾，大陆。而微软给出了一份更加http://msdn.microsoft.com/en-us/library/ms533052(v=vs.85).aspx">详细的语言列表，其中细分了zh-cn, zh-hk, zh-tw。

```
<!DOCTYPE html>

<html lang="en-us">

  ...

</html>
```

- **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，通常指定为'UTF-8'。

```
<!DOCTYPE html>

<html>

  <head>

    <meta charset="UTF-8">

  </head>

  ...

</html>
```

- **IE兼容模式**
    - 用 `` 标签可以指定页面应该用什么版本的IE来渲染;
    - 如果你想要了解更多，请点击这里;
    - 不同doctype在不同浏览器下会触发不同的渲染模式（这篇文章总结的很到位）。

```
<!DOCTYPE html>

<html>

  <head>

    <meta http-equiv="X-UA-Compatible" content="IE=Edge">

  </head>

  ...

</html>
```

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

- **HTML5 规范链接**
    - 使用link
    - 使用style
    - 使用script

```
<!-- External CSS -->

<link rel="stylesheet" href="code_guide.css">



<!-- In-document CSS -->

<style>

    ...

</style>



<!-- External JS -->

<script src="code_guide.js"></script>



<!-- In-document JS -->

<script>

    ...

</script>
```

- **属性顺序**
    - 属性应该按照特定的顺序出现以保证易读性；
        - class
        - id
        - name
        - data-*
        - src, for, type, href, value , max-length, max, min, pattern
        - placeholder, title, alt
        - aria-*, role
        - required, readonly, disabled
    - class是为高可复用组件设计的，所以应处在第一位；
    - id更加具体且应该尽量少使用，所以将它放在第二位。

```
<a class="..." id="..." data-modal="toggle" href="#">Example link</a>



<input class="form-control" type="text">



<img src="..." alt="...">
```

- **boolean属性**
    - boolean属性指不需要声明取值的属性，XHTML需要每个属性声明取值，但是HTML5并不需要；
    - 更多内容可以参考 WhatWG section on boolean attributes：
    - boolean属性的存在表示取值为true，不存在则表示取值为false。

```
<input type="text" disabled>



<input type="checkbox" value="1" checked>



<select>

    <option value="1" selected>1</option>

</select>
```

- **JS生成标签**
    - 在JS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。
    - 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。

```
<!-- Not well -->

<span class="avatar">

    <img src="...">

</span>



<!-- Better -->
```

```
<img class="avatar" src="...">
```

- **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。

CSS/SCSS

- **缩进**
    - 使用soft tab（4个空格）。

```
.element {
    position: absolute;
    top: 10px;
    left: 10px;

    border-radius: 10px;
    width: 50px;
    height: 50px;
}
```

- **分号**
    - 每个属性声明末尾都要加分号。

```
.element {
    width: 20px;
    height: 20px;

    background-color: red;
}
```

- **空格**
    - 以下几种情况不需要空格：

        - 属性名后
        - 多个规则的分隔符','前
        - !important '!'后
        - 属性值中'('后和')'前
        - 行末不要有多余的空格
    - 以下几种情况需要空格:
        - 属性值前
        - 选择器'>', '+', '~'前后
        - '{'前
        - !important '!'前
        - @else 前后
        - 属性值中的','后
        - 注释'/'后和'/'前

```
/* not good */

.element {

    color :red! important;

    background-color: rgba(0,0,0,.5);

}



/* good */

.element {

    color: red !important;

    background-color: rgba(0, 0, 0, .5);

}



/* not good */

.element ,

.dialog{

    ...

}
```

```
/* good */

.element,

.dialog {

    ...

}



/* not good */

.element>.dialog{

    ...

}



/* good */

.element > .dialog{

    ...

}



/* not good */

.element{

    ...

}



/* good */

.element {

    ...

}
```

```
/* not good */
@if{
    ...
}@else{
    ...
}

/* good */
@if {
    ...
} @else {
    ...
}
```

- **空行**
    - 以下几种情况需要空行：
        - 文件最后保留一个空行
        - '}'后最好跟一个空行，包括scss中嵌套的规则
        - 属性之间需要适当的空行，具体见属性声明顺序

```
/* not good */
.element {
    ...
}
.dialog {
    color: red;
    &:after {
        ...
```

```
    }
}

/* good */
.element {
    ...
}

.dialog {
    color: red;

    &:after {
        ...
    }
}
```

- **换行**
    - 以下几种情况不需要换行:
        - '{'前
    - 以下几种情况需要换行:
        - '{'后和'}'前
        - 每个属性独占一行
        - 多个规则的分隔符','后

```
/* not good */
.element
{color: red; background-color: black;}

/* good */
.element {
```

```
    color: red;
    background-color: black;
}

/* not good */
.element, .dialog {
    ...
}

/* good */
.element,
.dialog {
    ...
}
```

- **注释**
    - 注释统一用'/* */'（scss中也不要用'//'），具体参照右边的写法；
    - 缩进与下一行代码保持一致；
    - 可位于一个代码行的末尾，与代码间隔一个空格。

```
/* Modal header */
.modal-header {
    ...
}

/*
 * Modal header
 */
.modal-header {
```

```
    ...
}

.modal-header {
    /* 50px */
    width: 50px;

    color: red; /* color red */
}
```

- **引号**
    - 最外层统一使用双引号；
    - url的内容要用引号；
    - 属性选择器中的属性值需要引号。

```
.element:after {
    content: "";
    background-image: url("logo.png");
}

li[data-type="single"] {
    ...
}
```

- **命名**
    - 类名使用小写字母，以中划线分隔
    - id采用驼峰式命名
    - scss中的变量、函数、混合、placeholder采用驼峰式命名

```
/* class */
.element-content {
```

```
    ...
}

/* id */
#myDialog {
    ...
}

/* 变量 */
$colorBlack: #000;

/* 函数 */
@function pxToRem($px) {
    ...
}

/* 混合 */
@mixin centerBlock {
    ...
}

/* placeholder */
%myDialog {
    ...
}
```

- **属性声明顺序**
    - 相关的属性声明按右边的顺序做分组处理，组之间需要有一个空行。

```
.declaration-order
    display: block;
    float: right;

    position: absolute;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    z-index: 100;

    border: 1px solid #e5e5e5;
    border-radius: 3px;
    width: 100px;
    height: 100px;

    font: normal 13px "Helvetica Neue", sans-serif;
    line-height: 1.5;
    text-align: center;

    color: #333;
    background-color: #f5f5f5;

    opacity: 1;
```

```
}

// 下面是推荐的属性的顺序

[

    [

        "display",

        "visibility",

        "float",

        "clear",

        "overflow",

        "overflow-x",

        "overflow-y",

        "clip",

        "zoom"

    ],

    [

        "table-layout",

        "empty-cells",

        "caption-side",

        "border-spacing",

        "border-collapse",

        "list-style",

        "list-style-position",

        "list-style-type",

        "list-style-image"

    ],

    [
```

```
    "-webkit-box-orient",
    "-webkit-box-direction",
    "-webkit-box-decoration-break",
    "-webkit-box-pack",
    "-webkit-box-align",
    "-webkit-box-flex"
],
[
    "position",
    "top",
    "right",
    "bottom",
    "left",
    "z-index"
],
[
    "margin",
    "margin-top",
    "margin-right",
    "margin-bottom",
    "margin-left",
    "-webkit-box-sizing",
    "-moz-box-sizing",
    "box-sizing",
    "border",
```

```
"border-width",

"border-style",

"border-color",

"border-top",

"border-top-width",

"border-top-style",

"border-top-color",

"border-right",

"border-right-width",

"border-right-style",

"border-right-color",

"border-bottom",

"border-bottom-width",

"border-bottom-style",

"border-bottom-color",

"border-left",

"border-left-width",

"border-left-style",

"border-left-color",

"-webkit-border-radius",

"-moz-border-radius",

"border-radius",

"-webkit-border-top-left-radius",

"-moz-border-radius-topleft",

"border-top-left-radius",
```

"-webkit-border-top-right-radius",

"-moz-border-radius-topright",

"border-top-right-radius",

"-webkit-border-bottom-right-radius",

"-moz-border-radius-bottomright",

"border-bottom-right-radius",

"-webkit-border-bottom-left-radius",

"-moz-border-radius-bottomleft",

"border-bottom-left-radius",

"-webkit-border-image",

"-moz-border-image",

"-o-border-image",

"border-image",

"-webkit-border-image-source",

"-moz-border-image-source",

"-o-border-image-source",

"border-image-source",

"-webkit-border-image-slice",

"-moz-border-image-slice",

"-o-border-image-slice",

"border-image-slice",

"-webkit-border-image-width",

"-moz-border-image-width",

"-o-border-image-width",

"border-image-width",

```
    "-webkit-border-image-outset",
    "-moz-border-image-outset",
    "-o-border-image-outset",
    "border-image-outset",
    "-webkit-border-image-repeat",
    "-moz-border-image-repeat",
    "-o-border-image-repeat",
    "border-image-repeat",
    "padding",
    "padding-top",
    "padding-right",
    "padding-bottom",
    "padding-left",
    "width",
    "min-width",
    "max-width",
    "height",
    "min-height",
    "max-height"
],
[
    "font",
    "font-family",
    "font-size",
    "font-weight",
```

```
"font-style",
"font-variant",
"font-size-adjust",
"font-stretch",
"font-effect",
"font-emphasize",
"font-emphasize-position",
"font-emphasize-style",
"font-smooth",
"line-height",
"text-align",
"-webkit-text-align-last",
"-moz-text-align-last",
"-ms-text-align-last",
"text-align-last",
"vertical-align",
"white-space",
"text-decoration",
"text-emphasis",
"text-emphasis-color",
"text-emphasis-style",
"text-emphasis-position",
"text-indent",
"-ms-text-justify",
"text-justify",
```

```
    "letter-spacing",
    "word-spacing",
    "-ms-writing-mode",
    "text-outline",
    "text-transform",
    "text-wrap",
    "-ms-text-overflow",
    "text-overflow",
    "text-overflow-ellipsis",
    "text-overflow-mode",
    "-ms-word-wrap",
    "word-wrap",
    "-ms-word-break",
    "word-break"
],
[
    "color",
    "background",
    "filter:progid:DXImageTransform.Microsoft.AlphaImageLoader",
    "background-color",
    "background-image",
    "background-repeat",
    "background-attachment",
    "background-position",
    "-ms-background-position-x",
```

```
    "background-position-x",
    "-ms-background-position-y",
    "background-position-y",
    "-webkit-background-clip",
    "-moz-background-clip",
    "background-clip",
    "background-origin",
    "-webkit-background-size",
    "-moz-background-size",
    "-o-background-size",
    "background-size"
],
[
    "outline",
    "outline-width",
    "outline-style",
    "outline-color",
    "outline-offset",
    "opacity",
    "filter:progid:DXImageTransform.Microsoft.Alpha(Opacity",
    "-ms-filter:\\'progid:DXImageTransform.Microsoft.Alpha",
    "-ms-interpolation-mode",
    "-webkit-box-shadow",
    "-moz-box-shadow",
    "box-shadow",
```

```
    "filter:progid:DXImageTransform.Microsoft.gradient",
    "-ms-filter:\\'progid:DXImageTransform.Microsoft.gradient",
    "text-shadow"
],
[
    "-webkit-transition",
    "-moz-transition",
    "-ms-transition",
    "-o-transition",
    "transition",
    "-webkit-transition-delay",
    "-moz-transition-delay",
    "-ms-transition-delay",
    "-o-transition-delay",
    "transition-delay",
    "-webkit-transition-timing-function",
    "-moz-transition-timing-function",
    "-ms-transition-timing-function",
    "-o-transition-timing-function",
    "transition-timing-function",
    "-webkit-transition-duration",
    "-moz-transition-duration",
    "-ms-transition-duration",
    "-o-transition-duration",
    "transition-duration",
```

```
"-webkit-transition-property",

"-moz-transition-property",

"-ms-transition-property",

"-o-transition-property",

"transition-property",

"-webkit-transform",

"-moz-transform",

"-ms-transform",

"-o-transform",

"transform",

"-webkit-transform-origin",

"-moz-transform-origin",

"-ms-transform-origin",

"-o-transform-origin",

"transform-origin",

"-webkit-animation",

"-moz-animation",

"-ms-animation"

"-o-animation",

"animation",

"-webkit-animation-name",

"-moz-animation-name",

"-ms-animation-name",

"-o-animation-name",

"animation-name",
```

```
"-webkit-animation-duration",

"-moz-animation-duration",

"-ms-animation-duration",

"-o-animation-duration",

"animation-duration",

"-webkit-animation-play-state",

"-moz-animation-play-state",

"-ms-animation-play-state",

"-o-animation-play-state",

"animation-play-state",

"-webkit-animation-timing-function",

"-moz-animation-timing-function",

"-ms-animation-timing-function",

"-o-animation-timing-function",

"animation-timing-function",

"-webkit-animation-delay",

"-moz-animation-delay",

"-ms-animation-delay",

"-o-animation-delay",

"animation-delay",

"-webkit-animation-iteration-count",

"-moz-animation-iteration-count",

"-ms-animation-iteration-count",

"-o-animation-iteration-count",

"animation-iteration-count",
```

```
    "-webkit-animation-direction",
    "-moz-animation-direction",
    "-ms-animation-direction",
    "-o-animation-direction",
    "animation-direction"
],
[
    "content",
    "quotes",
    "counter-reset",
    "counter-increment",
    "resize",
    "cursor",
    "-webkit-user-select",
    "-moz-user-select",
    "-ms-user-select",
    "user-select",
    "nav-index",
    "nav-up",
    "nav-right",
    "nav-down",
    "nav-left",
    "-moz-tab-size",
    "-o-tab-size",
    "tab-size",
```

```
        "-webkit-hyphens",
        "-moz-hyphens",
        "hyphens",
        "pointer-events"
    ]
]
```

- **颜色**
    - 颜色16进制用小写字母；
    - 颜色16进制尽量用简写。

```
/* not good */
.element {
    color: #ABCDEF;
    background-color: #001122;
}

/* good */
.element {
    color: #abcdef;
    background-color: #012;
}
```

- **属性简写**
    - 属性简写需要你非常清楚属性值的正确顺序，而且在大多数情况下并不需要设置属性简写中包含的所有值，所以建议尽量分开声明会更加清晰；
    - margin 和 padding 相反，需要使用简写；
    - 常见的属性简写包括：
        - font
        - background
        - transition
        - animation

```
/* not good */
.element {
    transition: opacity 1s linear 2s;
}

/* good */
.element {
    transition-delay: 2s;
    transition-timing-function: linear;
    transition-duration: 1s;
    transition-property: opacity;
}
```

- **媒体查询**
    - 尽量将媒体查询的规则靠近与他们相关的规则，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。

```
.element {
    ...
}

.element-avatar{
    ...
}

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

```
    }


    .element-avatar {

        ...

    }

}
```

- **SCSS相关**
    - 提交的代码中不要有 @debug;
    - 声明顺序:
        - @extend
        - 不包含 @content 的 @include
        - 包含 @content 的 @include
        - 自身属性
        - 嵌套规则
    - @import 引入的文件不需要开头的'_'和结尾的'.scss';
    - 嵌套最多不能超过5层;
    - @extend 中使用placeholder选择器;
    - 去掉不必要的父级引用符号'&'。

```
/* not good */

@import "_dialog.scss";



/* good */

@import "dialog";



/* not good */

.fatal {

    @extend .error;

}



/* good */
```

```
.fatal {
    @extend %error;
}


/* not good */
.element {
    & > .dialog {
        ...
    }
}


/* good */
.element {
    > .dialog {
        ...
    }
}
```

- **杂项**
    - 不允许有空的规则；
    - 元素选择器用小写字母；
    - 去掉小数点前面的0；
    - 去掉数字中不必要的小数点和末尾的0；
    - 属性值'0'后面不要加单位；
    - 同个属性不同前缀的写法需要在垂直方向保持对齐，具体参照右边的写法；
    - 无前缀的标准属性应该写在有前缀的属性后面；
    - 不要在同个规则里出现重复的属性，如果重复的属性是连续的则没关系；
    - 不要在一个文件里出现两个相同的规则；
    - 用 border: 0; 代替 border: none;；
    - 选择器不要超过4层（在scss中如果超过4层应该考虑用嵌套的方式来写）；
    - 发布的代码中不要有 @import；
    - 尽量少用'*'选择器。

```
/* not good */

.element {

}



/* not good */

Ll {

    ...

}



/* good */

li {

    ...

}



/* not good */

.element {

    color: rgba(0, 0, 0, 0.5);

}



/* good */

.element {

    color: rgba(0, 0, 0, .5);

}



/* not good */

.element {
```

```
    width: 50.0px;
}

/* good */
.element {
    width: 50px;
}

/* not good */
.element {
    width: 0px;
}

/* good */
.element {
    width: 0;
}

/* not good */
.element {
    border-radius: 3px;
    -webkit-border-radius: 3px;
    -moz-border-radius: 3px;

    background: linear-gradient(to bottom, #fff 0, #eee 100%);
```

```
    background: -webkit-linear-gradient(top, #fff 0, #eee 100%);
    background: -moz-linear-gradient(top, #fff 0, #eee 100%);
}

/* good */
.element {
    -webkit-border-radius: 3px;
       -moz-border-radius: 3px;
            border-radius: 3px;

    background: -webkit-linear-gradient(top, #fff 0, #eee 100%);
    background:    -moz-linear-gradient(top, #fff 0, #eee 100%);
    background:         linear-gradient(to bottom, #fff 0, #eee 100%);
}

/* not good */
.element {
    color: rgb(0, 0, 0);
    width: 50px;
    color: rgba(0, 0, 0, .5);
}

/* good */
.element {
    color: rgb(0, 0, 0);
```

```
    color: rgba(0, 0, 0, .5);

}
```

JavaScript

- **缩进**
    - 使用soft tab（4个空格）。

```
var x = 1,

    y = 1;



if (x < y) {

    x += 10;

} else {

    x += 1;

}
```

- **单行长度**
    - 不要超过80，但如果编辑器开启word wrap可以不考虑单行长度。
- **分号**
    - 以下几种情况后需加分号：
        - 变量声明
        - 表达式
        - return
        - throw
        - break
        - continue
        - do-while

```
/* var declaration */

var x = 1;



/* expression statement */
```

```
x++;



/* do-while */

do {

    x++;

} while (x < 10);
```

- **空格**
    - 以下几种情况不需要空格:
        - 对象的属性名后
        - 前缀一元运算符后
        - 后缀一元运算符前
        - 函数调用括号前
        - 无论是函数声明还是函数表达式，'('前不要空格
        - 数组的'['后和']'前
        - 对象的'{'后和'}'前
        - 运算符'('后和')'前
    - 以下几种情况需要空格:
        - 二元运算符前后
        - 三元运算符'?:'前后
        - 代码块'{'前
        - 下列关键字前：else, while, catch, finally
        - 下列关键字后：if, else, for, while, do, switch, case, try,catch, finally, with, return, typeof
        - 单行注释'//'后（若单行注释和代码同行，则'//'前也需要），多行注释'*'后
        - 对象的属性值前
        - for循环，分号后留有一个空格，前置条件如果有多个，逗号后留一个空格
        - 无论是函数声明还是函数表达式，'{'前一定要有空格
        - 函数的参数之间

```
// not good

var a = {

    b :1

};



// good
```

```
var a = {
    b: 1
};


// not good
++ x;
y ++;
z = x?1:2;


// good
++x;
y++;
z = x ? 1 : 2;


// not good
var a = [ 1, 2 ];


// good
var a = [1, 2];


// not good
var a = ( 1+2 )*3;


// good
var a = (1 + 2) * 3;
```

```
// no space before '(', one space before '{', one space between function parameters

var doSomething = function(a, b, c) {

    // do something

};



// no space before '('

doSomething(item);



// not good

for(i=0;i<6;i++){

    x++;

}



// good

for (i = 0; i < 6; i++) {

    x++;

}
```

- **空行**
    - 以下几种情况需要空行：
        - 变量声明后（当变量声明在代码块的最后一行时，则无需空行）
        - 注释前（当注释在代码块的第一行时，则无需空行）
        - 代码块后（在函数调用、数组、对象中则无需空行）
        - 文件最后保留一个空行

```
// need blank line after variable declaration

var x = 1;
```

```
// not need blank line when variable declaration is last expression in the current block

if (x >= 1) {

    var y = x + 1;

}



var a = 2;



// need blank line before line comment

a++;



function b() {

    // not need blank line when comment is first line of block

    return a;

}



// need blank line after blocks

for (var i = 0; i < 2; i++) {

    if (true) {

        return false;

    }



    continue;

}



var obj = {
```

```
    foo: function() {
        return 1;
    },

    bar: function() {
        return 2;
    }
};

// not need blank line when in argument list, array, object
func(
    2,
    function() {
        a++;
    },
    3
);

var foo = [
    2,
    function() {
        a++;
    },
    3
];
```

```
var foo = {
    a: 2,
    b: function() {
        a++;
    },
    c: 3
};
```

- **换行**
    - 换行的地方，行末必须有','或者运算符;
    - 以下几种情况不需要换行:
        - 下列关键字后：else, catch, finally
        - 代码块'{'前
    - 以下几种情况需要换行:
        - 代码块'{'后和'}'前
        - 变量赋值后

```
// not good
var a = {
    b: 1
    , c: 2
};


x = y
    ? 1 : 2;


// good
var a = {
    b: 1,
```

```
    c: 2
};


x = y ? 1 : 2;
x = y ?
    1 : 2;


// no need line break with 'else', 'catch', 'finally'
if (condition) {
    ...
} else {
    ...
}


try {
    ...
} catch (e) {
    ...
} finally {
    ...
}


// not good
function test()
{
```

```
    ...
}


// good
function test() {
    ...
}


// not good
var a, foo = 7, b,
    c, bar = 8;


// good
var a,
    foo = 7,
    b, c, bar = 8;
```

- **换行**
    - 换行的地方，行末必须有','或者运算符;
    - 以下几种情况不需要换行:
        - 下列关键字后：else, catch, finally
        - 代码块'{'前
    - 以下几种情况需要换行:
        - 代码块'{'后和'}'前
        - 变量赋值后

```
// not good
var a = {
    b: 1
    , c: 2
```

```
};


x = y
    ? 1 : 2;


// good
var a = {
    b: 1,
    c: 2
};


x = y ? 1 : 2;
x = y ?
    1 : 2;


// no need line break with 'else', 'catch', 'finally'
if (condition) {
    ...
} else {
    ...
}


try {
    ...
} catch (e) {
```

```
    ...
} finally {
    ...
}


// not good
function test()
{
    ...
}


// good
function test() {
    ...
}


// not good
var a, foo = 7, b,
    c, bar = 8;


// good
var a,
    foo = 7,
    b, c, bar = 8;
```

- **单行注释**

    - 双斜线后，必须跟一个空格；
    - 缩进与下一行代码保持一致；
    - 可位于一个代码行的末尾，与代码间隔一个空格。

```
if (condition) {

    // if you made it here, then all security checks passed

    allowed();

}



var zhangsan = 'zhangsan'; // one space after code
```

- **多行注释**
    - 最少三行，'*'后跟一个空格，具体参照右边的写法；
    - 建议在以下情况下使用：
        - 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

```
/*

 * one space after '*'

 */

var x = 1;
```

- **文档注释**
    - 各类标签@param, @method等请参考usejsdoc和JSDoc Guide；
    - 建议在以下情况下使用：
        - 所有常量
        - 所有函数
        - 所有类

```
/**

 * @func

 * @desc 一个带参数的函数

 * @param {string} a - 参数a

 * @param {number} b=1 - 参数b默认值为1
```

```
 * @param {string} c=1 - 参数c有两种支持的取值</br>1—表示x</br>2—表示xx

 * @param {object} d - 参数d为一个对象

 * @param {string} d.e - 参数d的e属性

 * @param {string} d.f - 参数d的f属性

 * @param {object[]} g - 参数g为一个对象数组

 * @param {string} g.h - 参数g数组中一项的h属性

 * @param {string} g.i - 参数g数组中一项的i属性

 * @param {string} [j] - 参数j是一个可选参数

 */

function foo(a, b, c, d, g, j) {

    ...

}
```

- **引号**
    - 最外层统一使用单引号。

```
// not good

var x = "test";



// good

var y = 'foo',

    z = '<div id="test"></div>';
```

- **变量命名**
    - 标准变量采用驼峰式命名（除了对象的属性外，主要是考虑到cgi返回的数据）
    - 'ID'在变量名中全大写
    - 'URL'在变量名中全大写
    - 'Android'在变量名中大写第一个字母
    - 'iOS'在变量名中小写第一个，大写后两个字母
    - 常量全大写，用下划线连接
    - 构造函数，大写第一个字母
    - jquery对象必须以'$'开头命名

```
var thisIsMyName;



var goodID;



var reportURL;



var AndroidVersion;



var iOSVersion;



var MAX_COUNT = 10;



function Person(name) {

    this.name = name;

}



// not good

var body = $('body');



// good

var $body = $('body');
```

- **变量声明**
    - 一个函数作用域中所有的变量声明尽量提到函数首部，用一个var声明，不允许出现两个连续的var声明。

```
function doSomethingWithItems(items) {

    // use one var
```

```
    var value = 10,
        result = value + 10,
        i,
        len;

    for (i = 0, len = items.length; i < len; i++) {
        result += 10;
    }
}
```

- **函数**
    - 无论是函数声明还是函数表达式，'('前不要空格，但'{'前一定要有空格；
    - 函数调用括号前不需要空格；
    - 立即执行函数外必须包一层括号；
    - 不要给inline function命名；
    - 参数之间用', '分隔，注意逗号后有一个空格。

```
// no space before '(', but one space before'{'
var doSomething = function(item) {
    // do something
};

function doSomething(item) {
    // do something
}

// not good
doSomething (item);
```

```
// good

doSomething(item);



// requires parentheses around immediately invoked function expressions

(function() {

    return 1;

})();



// not good

[1, 2].forEach(function x() {

    ...

});



// good

[1, 2].forEach(function() {

    ...

});



// not good

var a = [1, 2, function a() {

    ...

}];



// good

var a = [1, 2, function() {
```

```
    ...
}];


// use ', ' between function parameters
var doSomething = function(a, b, c) {
    // do something
};
```

- **数组、对象**
    - 对象属性名不需要加引号;
    - 对象以缩进的形式书写，不要写在一行;
    - 数组、对象最后不要有逗号。

```
// not good
var a = {
    'b': 1
};


var a = {b: 1};


var a = {
    b: 1,
    c: 2,
};


// good
var a = {
    b: 1,
```

```
    c: 2

};
```

- **括号**
    - 下列关键字后必须有大括号（即使代码块的内容只有一行）：if, else,for, while, do, switch, try, catch, finally, with。

```
// not good

if (condition)

    doSomething();



// good

if (condition) {

    doSomething();

}
```

- **null**
    - 适用场景：
        - 初始化一个将来可能被赋值为对象的变量
        - 与已经初始化的变量做比较
        - 作为一个参数为对象的函数的调用传参
        - 作为一个返回对象的函数的返回值
    - 不适用场景：
        - 不要用null来判断函数调用时有无传参
        - 不要与未初始化的变量做比较

```
// not good

function test(a, b) {

    if (b === null) {

        // not mean b is not supply

        ...

    }

}
```

```
var a;



if (a === null) {

    ...

}



// good

var a = null;



if (a === null) {

    ...

}
```

- undefined
    - 永远不要直接使用undefined进行变量判断;
    - 使用typeof和字符串'undefined'对变量进行判断。

```
// not good

if (person === undefined) {

    ...

}



// good

if (typeof person === 'undefined') {

    ...

}
```

- jshint

- 用'===', '!=='代替'==', '!=';
- for-in里一定要有hasOwnProperty的判断;
- 不要在内置对象的原型上添加方法，如Array, Date;
- 不要在内层作用域的代码里声明了变量，之后却访问到了外层作用域的同名变量;
- 变量不要先使用后声明;
- 不要在一句代码中单单使用构造函数，记得将其赋值给某个变量;
- 不要在同个作用域下声明同名变量;
- 不要在一些不需要的地方加括号，例：delete(a.b);
- 不要使用未声明的变量（全局变量需要加到.jshintrc文件的globals属性里面）;
- 不要声明了变量却不使用;
- 不要在应该做比较的地方做赋值;
- debugger不要出现在提交的代码里;
- 数组中不要存在空元素;
- 不要在循环内部声明函数;
- 不要像这样使用构造函数，例：new function () { ... }, new Object;

```
// not good

if (a == 1) {

    a++;

}



// good

if (a === 1) {

    a++;

}



// good

for (key in obj) {

    if (obj.hasOwnProperty(key)) {

        // be sure that obj[key] belongs to the object and was not inherited

        console.log(obj[key]);

    }

}
```

```
// not good
Array.prototype.count = function(value) {
    return 4;
};

// not good
var x = 1;

function test() {
    if (true) {
        var x = 0;
    }

    x += 1;
}

// not good
function test() {
    console.log(x);

    var x = 1;
}

// not good
```

```
new Person();



// good

var person = new Person();



// not good

delete(obj.attr);



// good

delete obj.attr;



// not good

if (a = 10) {

    a++;

}



// not good

var a = [1, , , 2, 3];



// not good

var nums = [];



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

    (function(i) {

        nums[i] = function(j) {
```

```
            return i + j;

        };

    }(i));

}



// not good

var singleton = new function() {

    var privateVar;



    this.publicMethod = function() {

        privateVar = 1;

    };



    this.publicMethod2 = function() {

        privateVar = 2;

    };

};
```

- **杂项**
    - 不要混用tab和space；
    - 不要在一处使用多个tab或space；
    - 换行符统一用'LF'；
    - 对上下文this的引用只能使用'_this', 'that', 'self'其中一个来命名；
    - 行尾不要有空白字符；
    - switch的falling through和no default的情况一定要有注释特别说明；
    - 不允许有空的代码块。

```
// not good

var a   = 1;
```

```
function Person() {
    // not good
    var me = this;

    // good
    var _this = this;

    // good
    var that = this;

    // good
    var self = this;
}

// good
switch (condition) {
    case 1:
    case 2:
        ...
        break;
    case 3:
        ...
    // why fall through
    case 4
        ...
```

```
        break;

    // why no default

}



// not good with empty block

if (condition) {



}
```