# HTML规范

## 代码规范

### DOCTYPE 声明

一个DOCTYPE必须包含以下部分，并严格按照顺序出现：

1. A string that is an ASCII case-insensitive match for the string “<!DOCTYPE”.
2. One or more space characters.
3. A string that is an ASCII case-insensitive match for the string “html”.
4. Optionally, a DOCTYPE legacy string or an obsolete permitted DOCTYPE string (defined below).
5. Zero or more space characters.
6. A “>” (U+003E) character.
7. 一个ASCII字符串 “<!DOCTYPE” ，大小写不敏感
8. 一个或多个空白字符
9. 一个ASCII字符串”html”，大小写不敏感
10. 一个可选的历史遗留的DOCTYPE字符串 （[DOCTYPE legacy string](http://www.w3.org/TR/2014/REC-html5-20141028/syntax.html#doctype-legacy-string)），或者一个可选的已过时但被允许的DOCTYPE字符串 （[obsolete permitted DOCTYPE string](http://www.w3.org/TR/2014/REC-html5-20141028/syntax.html#obsolete-permitted-doctype-string)） 字符串
11. 一个或多个空白字符
12. 一个编码为 U+003E 的字符 “>”

**团队约定**

HTML文件必须加上 DOCTYPE 声明，并统一使用 HTML5 的文档声明：

|  |
| --- |
| <!DOCTYPE html> |

**更多关于 DOCTYPE声明**

[#The DOCTYPE](http://www.w3.org/TR/2014/REC-html5-20141028/syntax.html#the-doctype)

### 页面语言LANG

Lang属性的取值应该遵循互联网工程任务组–IETF（The Internet Engineering Task Force）制定的关于语言标签的文档 [BCP 47 - Tags for Identifying Languages](http://tools.ietf.org/html/bcp47)

**团队约定**

推荐使用属性值 cmn-Hans-CN（简体, 中国大陆），但是考虑浏览器和操作系统的兼容性，目前仍然使用 zh-CN 属性值

|  |
| --- |
| <html lang="zh-CN"> |

更多地区语言参考：

zh-SG 中文 (简体, 新加坡) 对应 cmn-Hans-SG 普通话 (简体, 新加坡)

zh-HK 中文 (繁体, 香港) 对应 cmn-Hant-HK 普通话 (繁体, 香港)

zh-MO 中文 (繁体, 澳门) 对应 cmn-Hant-MO 普通话 (繁体, 澳门)

zh-TW 中文 (繁体, 台湾) 对应 cmn-Hant-TW 普通话 (繁体, 台湾)

**已废弃不推荐使用的 Languages Tags**

以下写法已于 2009 年废弃，请勿使用（cmn、wuu、yue、gan 等已由 2005 年的 extlang 升级到 2009 年的 language）：

zh-cmn, zh-cmn-Hans, zh-cmn-Hant, zh-wuu, zh-yue, zh-gan

以下写法已于 2009 年废弃，不推荐使用：

zh-Hans, zh-Hans-CN, zh-Hans-SG, zh-Hans-HK, zh-Hans-MO, zh-Hans-TW,

zh-Hant, zh-Hant-CN, zh-Hant-SG, zh-Hant-HK, zh-Hant-MO, zh-Hant-TW

更多已废弃 Languages Tags 参考 [IANA Language Subtag Registry](http://www.iana.org/assignments/language-subtag-registry/language-subtag-registry) 里面的 “Type: redundant“”

**更多关于 Languages Tags ：**

[W3C Language tags in HTML and XML](http://www.w3.org/International/articles/language-tags/)

[网页头部的声明应该是用 lang=”zh” 还是 lang=”zh-cn”？](http://www.zhihu.com/question/20797118?utm_source=weibo&utm_medium=weibo_share&utm_content=share_question&utm_campaign=share_sidebar)

### CHARSET

Because the character sets in ISO-8859 was limited in size, and not compatible in multilingual environments, the Unicode Consortium developed the Unicode Standard.

The Unicode Standard covers (almost) all the characters, punctuations, and symbols in the world.

Unicode enables processing, storage, and transport of text independent of platform and language.

The default character encoding in HTML-5 is UTF-8.

因为 ISO-8859 中字符集大小是有限的，且在多语言环境中不兼容，所以 Unicode 联盟开发了 Unicode 标准。

Unicode 标准覆盖了（几乎）所有的字符、标点符号和符号。

Unicode 使文本的处理、存储和运输，独立于平台和语言。

HTML-5 中默认的字符编码是 UTF-8

参阅 [HTML Unicode (UTF-8) Reference](http://www.w3schools.com/charsets/ref_html_utf8.asp)

**团队约定**

一般情况下统一使用 “UTF-8” 编码

|  |
| --- |
| <meta charset="UTF-8"> |

由于历史原因，有些业务可能会使用 “GBK” 编码

|  |
| --- |
| <meta charset="GBK"> |

请尽量统一写成标准的 “UTF-8”，不要写成 “utf-8” 或 “utf8” 或 “UTF8”。根据 [IETF对UTF-8的定义](http://www.ietf.org/rfc/rfc3629)，其编码标准的写法是 “UTF-8”；而 UTF8 或 utf8 的写法只是出现在某些编程系统中，如 .NET framework 的类 System.Text.Encoding 中的一个属性名就叫 UTF8。

**更多关于**

UTF-8写法: [UTF8 or UTF-8?](http://stackoverflow.com/questions/809620/utf8-or-utf-8)

GBK：[Application of IANA Charset Registration for GBK](http://www.ietf.org/assignments/charset-reg/GBK)

Charset ：[character-encoding-declaration](http://www.w3.org/TR/html5/document-metadata.html#character-encoding-declaration)

### 元素及标签闭合

HTML元素共有以下5种：

* 空元素：area、base、br、col、command、embed、hr、img、input、keygen、link、meta、param、source、track、wbr
* 原始文本元素：script、style
* RCDATA元素：textarea、title
* 外来元素：来自MathML命名空间和SVG命名空间的元素。
* 常规元素：其他HTML允许的元素都称为常规元素。

元素标签的闭合应遵循以下原则：

Tags are used to delimit the start and end of elements in the markup. Raw text, escapable raw text, and normal elements have a start tag to indicate where they begin, and an end tag to indicate where they end. The start and end tags of certain normal elements can be omitted, as described below in the section on optional tags. Those that cannot be omitted must not be omitted. Void elements only have a start tag; end tags must not be specified for void elements. Foreign elements must either have a start tag and an end tag, or a start tag that is marked as self-closing, in which case they must not have an end tag.

* 原始文本元素、RCDATA元素以及常规元素都有一个开始标签来表示开始，一个结束标签来表示结束。
* [某些元素的开始和结束标签是可以省略的](http://www.w3.org/TR/html5/syntax.html#optional-tags)，如果规定标签不能被省略，那么就绝对不能省略它。
* 空元素只有一个开始标签，且不能为空元素设置结束标签。
* 外来元素可以有一个开始标签和配对的结束标签，或者只有一个自闭合的开始标签，且后者情况下该元素不能有结束标签。

**团队约定**

为了能让浏览器更好的解析代码以及能让代码具有更好的可读性，有如下约定：

* 所有具有开始标签和结束标签的元素都要写上起止标签，某些允许省略开始标签或和束标签的元素亦都要写上。
* 空元素标签都不加 “/” 字符

*推荐：*

|  |
| --- |
| <div>  <h1>我是h1标题</h1>  <p>我是一段文字，我有始有终，浏览器能正确解析</p>  </div>    <br> |

*不推荐：*

|  |
| --- |
| <div>  <h1>我是h1标题</h1>  <p>我是一段文字，我有始无终，浏览器亦能正确解析  </div>  <br/> |

更多关于元素及标签关闭：[#Elements](http://www.w3.org/TR/html5/syntax.html#elements-0)

### 书写风格

**HTML代码大小写**

HTML标签名、类名、标签属性和大部分属性值统一用小写

*推荐：*

|  |
| --- |
| <div class="demo"></div> |

*不推荐：*

|  |
| --- |
| <div class="DEMO"></div>    <DIV CLASS="DEMO"></DIV> |

HTML文本、CDATA、JavaScript、meta标签某些属性等内容可大小写混合

|  |
| --- |
| <!-- 优先使用 IE 最新版本和 Chrome Frame -->  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"/>  <!-- HTML文本内容 -->  <h1>I AM WHAT I AM </h1>  <!-- JavaScript 内容 -->  <script type="text/javascript">  var demoName = 'demoName';  ...  </script>    <!-- CDATA 内容 -->  <script type="text/javascript"><![CDATA[  ...  ]]></script> |

**类型属性**

不需要为 CSS、JS 指定类型属性，HTML5 中默认已包含

*推荐：*

|  |
| --- |
| <link rel="stylesheet" href="" >  <script src=""></script> |

*不推荐：*

|  |
| --- |
| <link rel="stylesheet" type="text/css" href="" >  <script type="text/javascript" src="" ></script> |

**元素属性**

* 元素属性值使用双引号语法
* 元素属性值可以写上的都写上

*推荐：*

|  |
| --- |
| <input type="text">    <input type="radio" name="name" checked="checked" > |

*不推荐：*

|  |
| --- |
| <input type=text>  <input type='text'>    <input type="radio" name="name" checked > |

更多关于元素属性：[#Attributes](http://www.w3.org/TR/html5/syntax.html#attributes-0)

**特殊字符引用**

In certain cases described in other sections, text may be mixed with character references. These can be used to escape characters that couldn’t otherwise legally be included in text.

文本可以和字符引用混合出现。这种方法可以用来转义在文本中不能合法出现的字符。

在 HTML 中不能使用小于号 “<” 和大于号 “>”特殊字符，浏览器会将它们作为标签解析，若要正确显示，在 HTML 源代码中使用字符实体

*推荐：*

|  |
| --- |
| <a href="#">more&gt;&gt;</a> |

*不推荐：*

|  |
| --- |
| <a href="#">more>></a> |

更多关于符号引用：[#Character references](http://www.w3.org/TR/html5/syntax.html#character-references)

**代码缩进**

统一使用四个空格进行代码缩进，使得各编辑器表现一致（各编辑器有相关配置）

|  |
| --- |
| <div class="jdc">  <a href="#"></a>  </div> |

**纯数字输入框**

使用 type="tel" 而不是 type="number"

|  |
| --- |
| <input type="tel"> |

**代码嵌套**

元素嵌套规范，每个块状元素独立一行，内联元素可选

*推荐：*

|  |
| --- |
| <div>  <h1></h1>  <p></p>  </div>  <p><span></span><span></span></p> |

*不推荐：*

|  |
| --- |
| <div>  <h1></h1><p></p>  </div>  <p>  <span></span>  <span></span>  </p> |

段落元素与标题元素只能嵌套内联元素

*推荐：*

|  |
| --- |
| <h1><span></span></h1>  <p><span></span><span></span></p> |

*不推荐：*

|  |
| --- |
| <h1><div></div></h1>  <p><div></div><div></div></p> |

## 注释规范

**遵循标准**

HTML注释规范写法应该遵循以下标准：

Comments must start with the four character sequence U+003C LESS-THAN SIGN, U+0021 EXCLAMATION MARK, U+002D HYPHEN-MINUS, U+002D HYPHEN-MINUS (<!–). Following this sequence, the comment may have text, with the additional restriction that the text must not start with a single “>” (U+003E) character, nor start with a U+002D HYPHEN-MINUS character (-) followed by a “>” (U+003E) character, nor contain two consecutive U+002D HYPHEN-MINUS characters (–), nor end with a U+002D HYPHEN-MINUS character (-). Finally, the comment must be ended by the three character sequence U+002D HYPHEN-MINUS, U+002D HYPHEN-MINUS, U+003E GREATER-THAN SIGN (–>).

* 必须以4个有序字符开始：编码为 U+003C LESS-THAN SIGN 的小于号, 编码为 U+0021 EXCLAMATION MARK 的感叹号, 编码为 U+002D HYPHEN-MINUS 横线, 编码为 U+002D HYPHEN-MINUS横线 ，即 “<!–”
* 在此之后是注释内容，注释的内容有以下限制：
  + 不能以单个 “>” (U+003E) 字符开始
  + 不能以由 “-“（U+002D HYPHEN-MINUS）和 ”>” (U+003E) 组合的字符开始，即 “->”
  + 不能包含两个连续的 U+002D HYPHEN-MINUS 字符，即 “–”
  + 不能以一个 U+002D HYPHEN-MINUS 字符结束，即 “-”
* 必须以3个有序字符结束：U+002D HYPHEN-MINUS, U+002D HYPHEN-MINUS, U+003E GREATER-THAN SIGN，即 “–>”

标准写法：

|  |
| --- |
| <!--Comment Text--> |

错误的写法：

|  |
| --- |
| <!-->The Wrong Comment Text-->  <!--->The Wrong Comment Text-->  <!--The--Wrong--Comment Text-->  <!--The Wrong Comment Text---> |

参考 www.w3.org [#Comments](http://www.w3.org/TR/2014/REC-html5-20141028/syntax.html#comments)

**团队约定**

### 单行注释

一般用于简单的描述，如某些状态描述、属性描述等

注释内容前后各一个空格字符，注释位于要注释代码的上面，单独占一行

*推荐：*

|  |
| --- |
| <!-- Comment Text -->  <div>...</div> |

*不推荐：*

|  |
| --- |
| <div>...</div><!-- Comment Text -->    <div><!-- Comment Text -->  ...  </div> |

### 模块注释

一般用于描述模块的名称以及模块开始与结束的位置

注释内容前后各一个空格字符，<!-- S Comment Text --> 表示模块开始，<!-- E Comment Text --> 表示模块结束，模块与模块之间相隔一行

*推荐写法：*

|  |
| --- |
| <!-- S Comment Text A -->  <div class="mod\_a">  ...  </div>  <!-- E Comment Text A -->    <!-- S Comment Text B -->  <div class="mod\_b">  ...  </div>  <!-- E Comment Text B --> |

*不推荐写法：*

|  |
| --- |
| <!-- S Comment Text A -->  <div class="mod\_a">  ...  </div>  <!-- E Comment Text A -->  <!-- S Comment Text B -->  <div class="mod\_b">  ...  </div>  <!-- E Comment Text B --> |

### 嵌套模块注释

当模块注释内再出现模块注释的时候，为了突出主要模块，嵌套模块不再使用

|  |
| --- |
| <!-- S Comment Text -->  <!-- E Comment Text --> |

而改用

|  |
| --- |
| <!-- /Comment Text --> |

注释写在模块结尾标签底部，单独一行。

|  |
| --- |
| <!-- S Comment Text A -->  <div class="mod\_a">    <div class="mod\_b">  ...  </div>  <!-- /mod\_b -->    <div class="mod\_c">  ...  </div>  <!-- /mod\_c -->    </div>  <!-- E Comment Text A --> |

## 文件模版

HTML模版指的是团队使用的初始化HTML文件，里面会根据不同平台而采用不一样的设置，一般主要不同的设置就是 mata 标签的设置，以下是 PC 和移动端的 HTML 模版。

### HTML5标准模版

|  |
| --- |
| <!DOCTYPE html>  <html lang="zh-CN">  <head>  <meta charset="UTF-8">  <title>HTML5标准模版</title>  </head>  <body>    </body>  </html> |

团队约定

### 移动端

|  |
| --- |
| <!DOCTYPE html>  <html lang="zh-CN">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no, shrink-to-fit=no" >  <meta name="format-detection" content="telephone=no" >  <title>移动端HTML模版</title>    <!-- S DNS预解析 -->  <link rel="dns-prefetch" href="">  <!-- E DNS预解析 -->  <!-- S 线上样式页面片，开发请直接取消注释引用 -->  <!-- #include virtual="" -->  <!-- E 线上样式页面片 -->  <!-- S 本地调试，根据开发模式选择调试方式，请开发删除 -->  <link rel="stylesheet" href="css/index.css" >  <!-- /本地调试方式 -->  <link rel="stylesheet" href="http://srcPath/index.css" >  <!-- /开发机调试方式 -->  <!-- E 本地调试 -->  </head>  <body>  </body>  </html> |

### PC端

|  |
| --- |
| <!DOCTYPE html>  <html lang="zh-CN">  <head>  <meta charset="UTF-8">  <meta name="keywords" content="your keywords">  <meta name="description" content="your description">  <meta name="author" content="author,email address">  <meta name="robots" content="index,follow">  <meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1">  <meta name="renderer" content="ie-stand">  <title>PC端HTML模版</title>  <!-- S DNS预解析 -->  <link rel="dns-prefetch" href="">  <!-- E DNS预解析 -->  <!-- S 线上样式页面片，开发请直接取消注释引用 -->  <!-- #include virtual="" -->  <!-- E 线上样式页面片 -->  <!-- S 本地调试，根据开发模式选择调试方式，请开发删除 -->  <link rel="stylesheet" href="css/index.css" >  <!-- /本地调试方式 -->  <link rel="stylesheet" href="http://srcPath/index.css" >  <!-- /开发机调试方式 -->  <!-- E 本地调试 -->  </head>  <body>  </body>  </html> |

# CSS规范

## 编码规范

CSS样式表是一个序列通用字符集，传输和存储过程中，这些字符必须由支持 US-ASCII（例如 UTF-8, ISO 8859-x, SHIFT JIS 等）字符编码方式编译

**文档内嵌样式表编码**

When a style sheet is embedded in another document, such as in the STYLE element or “style” attribute of HTML, the style sheet shares the character encoding of the whole document.

当样式出现在其它文档，如 HTML 的 STYLE 标签或标签属性 “style”，样式的编码由文档的决定。

**文档外链样式表编码**

When a style sheet resides in a separate file, user agents must observe the following priorities when determining a style sheet’s character encoding (from highest priority to lowest):

1. An HTTP “charset” parameter in a “Content-Type” field (or similar parameters in other protocols)
2. BOM and/or @charset
3. or other metadata from the linking mechanism (if any)
4. charset of referring style sheet or document (if any)
5. Assume UTF-8

文档外链样式表的编码可以由以下各项按照由高到低的优先级顺序决定：

1. HTTP “Content-Type” 字段参数 “charset”（或其它协议相似的参数）
2. BOM（byte-order mark）和（或）@charset
3. Link 中的元数据设置（如果有的话）
4. 引用样式表字符集或文档编码（如果有的话）
5. 假定为 UTF-8 编码

**样式表编码**

Authors using an @charset rule must place the rule at the very beginning of the style sheet, preceded by no characters. (If a byte order mark is appropriate for the encoding used, it may precede the @charset rule.)

@charset must be written literally, i.e., the 10 characters ‘@charset “‘ (lowercase, no backslash escapes), followed by the encoding name, followed by ‘“;’.

* @charset规则一定要在样式文件的第一行首个字符位置开始，否则的话就会有机会让 BOM 设置生效（如果有设置 BOM 的话）而优于 @charset 作为样式表的编码
* @charset ""; 一定要写上，并且用小写字母，不能出现转义符

**团队约定**

* 样式文件必须写上 @charset 规则，并且一定要在样式文件的第一行首个字符位置开始写，编码名用 “UTF-8”
* 字符 @charset “”; 都用小写字母，不能出现转义符，编码名允许大小混写
* 考虑到在使用“UTF-8”编码情况下 BOM 对代码的污染和编码显示的问题，在可控范围下，坚决不使用 BOM。（更多关于 BOM 可参考 [BOM的介绍](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83%E7%B5%84%E9%A0%86%E5%BA%8F%E8%A8%98%E8%99%9F) 和 [「带 BOM 的 UTF-8」和「无 BOM 的 UTF-8」有什么区别？](http://www.zhihu.com/question/20167122) ）

*推荐：*

|  |
| --- |
| @charset "UTF-8";  .jdc{} |

*不推荐：*

|  |
| --- |
| /\*\*  \* @desc File Info  \* @author Author Name  \* @date 2015-10-10  \*/    /\* @charset规则不在文件首行首个字符开始 \*/  @charset "UTF-8";  .jdc{} |

|  |
| --- |
| @CHARSET "UTF-8";  /\* @charset规则没有用小写 \*/  .jdc{} |

|  |
| --- |
| /\* 无@charset规则 \*/  .jdc{} |

更多关于样式编码：[CSS style sheet representation](http://www.w3.org/TR/2011/REC-CSS2-20110607/syndata.html#charset)

**代码风格**

**代码格式化**

样式书写一般有两种：一种是紧凑格式 (Compact)

|  |
| --- |
| .jdc{ display: block;width: 50px;} |

一种是展开格式（Expanded）

|  |
| --- |
| .jdc{  display: block;  width: 50px;  } |

**团队约定**

统一使用展开格式书写样式

**代码大小写**

样式选择器，属性名，属性值关键字全部使用小写字母书写，属性字符串允许使用大小写。

|  |
| --- |
| /\* 推荐 \*/  .jdc{  display:block;  }    /\* 不推荐 \*/  .JDC{  DISPLAY:BLOCK;  } |

**选择器**

* 尽量少用通用选择器 \*
* 不使用 ID 选择器
* 不使用无具体语义定义的标签选择器

|  |
| --- |
| /\* 推荐 \*/  .jdc {}  .jdc li {}  .jdc li p{}  /\* 不推荐 \*/  \*{}  #jdc {}  .jdc div{} |

**代码缩进**

统一使用四个空格进行代码缩进，使得各编辑器表现一致（各编辑器有相关配置）

|  |
| --- |
| .jdc {  width: 100%;  height: 100%;  } |

**分号**

每个属性声明末尾都要加分号；

|  |
| --- |
| .jdc {  width: 100%;  height: 100%;  } |

**代码易读性**

左括号与类名之间一个空格，冒号与属性值之间一个空格

*推荐：*

|  |
| --- |
| .jdc {  width: 100%;  } |

*不推荐：*

|  |
| --- |
| .jdc{  width:100%;  } |

逗号分隔的取值，逗号之后一个空格

*推荐：*

|  |
| --- |
| .jdc {  box-shadow: 1px 1px 1px #333, 2px 2px 2px #ccc;  } |

*不推荐：*

|  |
| --- |
| .jdc {  box-shadow: 1px 1px 1px #333,2px 2px 2px #ccc;  } |

为单个css选择器或新申明开启新行

*推荐：*

|  |
| --- |
| .jdc,  .jdc\_logo,  .jdc\_hd {  color: #ff0;  }  .nav{  color: #fff;  } |

*不推荐：*

|  |
| --- |
| .jdc,jdc\_logo,.jdc\_hd {  color: #ff0;  }.nav{  color: #fff;  } |

颜色值 rgb() rgba() hsl() hsla() rect() 中不需有空格，且取值不要带有不必要的 0

*推荐：*

|  |
| --- |
| .jdc {  color: rgba(255,255,255,.5);  } |

*不推荐：*

|  |
| --- |
| .jdc {  color: rgba( 255, 255, 255, 0.5 );  } |

属性值十六进制数值能用简写的尽量用简写

*推荐：*

|  |
| --- |
| .jdc {  color: #fff;  } |

*不推荐：*

|  |
| --- |
| .jdc {  color: #ffffff;  } |

不要为 0 指明单位

*推荐：*

|  |
| --- |
| .jdc {  margin: 0 10px;  } |

*不推荐：*

|  |
| --- |
| .jdc {  margin: 0px 10px;  } |

**属性值引号**

css属性值需要用到引号时，统一使用单引号

|  |
| --- |
| /\* 推荐 \*/  .jdc {  font-family: 'Hiragino Sans GB';  }  /\* 不推荐 \*/  .jdc {  font-family: "Hiragino Sans GB";  } |

**属性书写顺序**

建议遵循以下顺序：

1. 布局定位属性：display / position / float / clear / visibility / overflow
2. 自身属性：width / height / margin / padding / border / background
3. 文本属性：color / font / text-decoration / text-align / vertical-align / white- space / break-word
4. 其他属性（CSS3）：content / cursor / border-radius / box-shadow / text-shadow / background:linear-gradient …

|  |
| --- |
| .jdc {  display: block;  position: relative;  float: left;  width: 100px;  height: 100px;  margin: 0 10px;  padding: 20px 0;  font-family: Arial, 'Helvetica Neue', Helvetica, sans-serif;  color: #333;  background: rgba(0,0,0,.5);  -webkit-border-radius: 10px;  -moz-border-radius: 10px;  -o-border-radius: 10px;  -ms-border-radius: 10px;  border-radius: 10px;  } |

[mozilla官方属性顺序推荐](https://www.mozilla.org/css/base/content.css)

**CSS3浏览器私有前缀写法**

CSS3 浏览器私有前缀在前，标准前缀在后

|  |
| --- |
| .jdc {  -webkit-border-radius: 10px;  -moz-border-radius: 10px;  -o-border-radius: 10px;  -ms-border-radius: 10px;  border-radius: 10px;  } |

更多关于浏览器私有前辍写法：[#Vendor-specific extensions](http://www.w3.org/TR/2011/REC-CSS2-20110607/syndata.html#vendor-keywords)

## 注释规范

Comments begin with the characters /\* and end with the characters \*/. They may occur anywhere outside other tokens, and their contents have no influence on the rendering. Comments may not be nested.

* 注释以字符 /\* 开始，以字符 \*/ 结束
* 注释不能嵌套

|  |
| --- |
| /\*Comment Text\*/ |

### 团队约定

#### 单行注释

注释内容第一个字符和最后一个字符都是一个空格字符，单独占一行，行与行之间相隔一行

推荐：

|  |
| --- |
| /\* Comment Text \*/  .jdc{}  /\* Comment Text \*/  .jdc{} |

不推荐：

|  |
| --- |
| /\*Comment Text\*/  .jdc{  display: block;  }  .jdc{  display: block;/\*Comment Text\*/  } |

#### 模块注释

注释内容第一个字符和最后一个字符都是一个空格字符，/\* 与 模块信息描述占一行，多个横线分隔符-与\*/占一行，行与行之间相隔两行

推荐：

|  |
| --- |
| /\* Module A  ---------------------------------------------------------------- \*/  .mod\_a {}  /\* Module B  ---------------------------------------------------------------- \*/  .mod\_b {} |

不推荐：

|  |
| --- |
| /\* Module A ---------------------------------------------------- \*/  .mod\_a {}  /\* Module B ---------------------------------------------------- \*/  .mod\_b {} |

#### 文件信息注释

在样式文件编码声明 @charset 语句下面注明页面名称、作者、创建日期等信息

|  |
| --- |
| @charset "UTF-8";  /\*\*  \* @desc File Info  \* @author Author Name  \* @date 2015-10-10  \*/ |

更多关于CSS注释：[#Comments](http://www.w3.org/TR/2011/REC-CSS2-20110607/syndata.html#comments)

## SASS规范（LESS参考）

## 语法选用

SASS有两种语法格式，一种是 SCSS (Sassy CSS)，另一种是缩进格式（Indented Syntax），有时称之为 Sass。

### SCSS

SCSS语法基于 CSS 语法扩展，每一个有效的 CSS 文件都是一个有效的具有相同含义的 SCSS 文件，换种说法就是 SCSS 能识别大多数的 CSS hacks 写法和浏览器前缀写法以及早期的 IE 滤镜写法，这种格式以 .scss 作为扩展名。

### Sass

Sass 使用 “缩进” 代替 “花括号” 表示属性属于某个选择器，用 “换行” 代替 “分号” 分隔属性，很多人认为这样做比 SCSS 更容易阅读，书写也更快速。缩进格式也可以使用 Sass 的全部功能，只是与 SCSS 相比个别地方采取了不同的表达方式，具体请查看 [the indented syntax reference](http://sass-lang.com/documentation/file.INDENTED_SYNTAX.html)。这种格式以 .sass 作为拓展名。

更详细的用法请阅读 SASS 官网文档：[DOCUMENTATION](http://sass-lang.com/documentation/file.SASS_REFERENCE.html)

### 团队约定

考虑到 SCSS 语法对 CSS 语法友好的兼容性和扩展性，我们在使用 SASS 编写样式的时候，统一使用 SCSS 语法

## 编码格式

When running on Ruby 1.9 and later, Sass is aware of the character encoding of documents. Sass follows the CSS spec to determine the encoding of a stylesheet, and falls back to the Ruby string encoding. This means that it first checks the Unicode byte order mark, then the @charset declaration, then the Ruby string encoding. If none of these are set, it will assume the document is in UTF-8.

当在 Ruby1.9或更新的版本运行的时候，SASS 能识辨文档的字符编码。SASS 遵循 CSS 规范去确定样式文件的编码，进而转回 Ruby 字符串编码。这意味着SASS编译的时候会首先检测 BOM，然后到 @charset 声明，再到 Ruby 字符串编码，如果以上都没有设置，SASS 会认为文档的编码为 UTF-8

### 团队约定

严格遵守上面 “CSS规范” 中的 [“编码规范”](https://guide.aotu.io/docs/css/code.html)

更多关于 SASS 编码：[SASS Encodings](http://sass-lang.com/documentation/file.SASS_REFERENCE.html)

## SASS注释规范

SASS支持 CSS 标准的多行注释 /\* \*/，同时也支持单行注释 //。

* 多行注释在使用非 Compressed 模式进行编译后的输出文件中会保留下来，单行注释 // 侧会被移除
* 多行注释和单行注释在 SASS 编译后输出的压缩 CSS 文件都会被移除
* 当多行注释内容第一个字符是感叹号 “!” 的时候，即 /\*! \*/，SASS 无论用哪一种编译方式编译注释都会保留
* 注释内容可以加入 SASS 变量

### 团队约定

SCSS 文件内

* 全部遵循 CSS 注释规范
* 不使用 /\*! \*/ 注释方式
* 注释内不放 SASS 变量

标准的注释规范如下：

|  |
| --- |
| @charset "UTF-8";  /\*\*  \* @desc File Info  \* @author liqinuo  \* @date 2015-10-10  \*/  /\* Module A  ----------------------------------------------------------------\*/  .mod\_a {}  /\* module A logo \*/  .mod\_a\_logo {}  /\* module A nav \*/  .mod\_a\_nav {}  /\* Module B  ----------------------------------------------------------------\*/  .mod\_b {}  /\* module B logo \*/  .mod\_b\_logo {}  /\* module B nav \*/  .mod\_b\_nav {} |

## 嵌套规范

### 选择器嵌套

|  |
| --- |
| /\* CSS \*/  .jdc {}  body .jdc {}  /\* SCSS \*/  .jdc {  body & {}  } |

|  |
| --- |
| /\* CSS \*/  .jdc {}  .jdc\_cover {}  .jdc\_info {}  .jdc\_info\_name {}  /\* SCSS \*/  .jdc {  &\_cover {}  &\_info {  &\_name {}  }  } |

### 属性嵌套

|  |
| --- |
| /\* CSS \*/  .jdc {  background-color: red;  background-repeat: no-repeat;  background-image: url(/img/icon.png);  background-position: 0 0;  }  /\* SCSS \*/  .jdc {  background: {  color: red;  repeat: no-repeat;  image: url(/img/icon.png);  position: 0 0;  }  } |

## 变量

可复用属性尽量抽离为页面变量，易于统一维护

|  |
| --- |
| // CSS  .jdc {  color: red;  border-color: red;  }  // SCSS  $color: red;  .jdc {  color: $color;  border-color: $color;  } |

## 混合(mixin)

根据功能定义模块，然后在需要使用的地方通过 @include 调用，避免编码时重复输入代码段

|  |
| --- |
| // CSS  .jdc\_1 {  -webkit-border-radius: 5px;  border-radius: 5px;  }  .jdc\_2 {  -webkit-border-radius: 10px;  border-radius: 10px;  }  // SCSS  @mixin radius($radius:5px) {  -webkit-border-radius: $radius;  border-radius: $radius;  }  .jdc\_1 {  @include radius; //参数使用默认值  }  .jdc\_2 {  @include radius(10px);  }  // CSS  .jdc\_1 {  background: url(/img/icon.png) no-repeat -10px 0;  }  .jdc\_2 {  background: url(/img/icon.png) no-repeat -20px 0;  }  // SCSS  @mixin icon($x:0, $y:0) {  background: url(/img/icon.png) no-repeat $x, $y;  }  .jdc\_1 {  @include icon(-10px, 0);  }  .jdc\_2 {  @include icon(-20px, 0);  } |

## 占位选择器 %

如果不调用则不会有任何多余的 css 文件，占位选择器以 % 标识定义，通过 @extend 调用

|  |
| --- |
| //scss  %borderbox {  -webkit-box-sizing: border-box;  box-sizing: border-box;  }  .jdc {  @extend %borderbox;  } |

## extend 继承

|  |
| --- |
| // CSS  .jdc\_1 {  font-size: 12px;  color: red;  }  .jdc\_2 {  font-size: 12px;  color: red;  font-weight: bold;  }  // SCSS  .jdc\_1 {  font-size: 12px;  color: red;  }  .jdc\_2 {  @extend .jdc\_1;  font-weight: bold;  }  // 或者  %font\_red {  font-size: 12px;  color: red;  }  .jdc\_1 {  @extend %font\_red;  }  .jdc\_2 {  @extend %font\_red;  font-weight: bold;  } |

## for 循环

|  |
| --- |
| // CSS  .jdc\_1 {background-position: 0 -20px;}  .jdc\_2 {background-position: 0 -40px;}  .jdc\_3 {background-position: 0 -60px;}  // SCSS  @for $i from 1 through 3 {  .jdc\_#{$i} {  background-position: 0 (-20px) \* $i;  }  } |

注意：#{} 是连接符，变量连接使用时需要依赖

## each 循环

|  |
| --- |
| // CSS  .jdc\_list {  background-image: url(/img/jdc\_list.png);  }  .jdc\_detail {  background-image: url(/img/jdc\_detail.png);  }  // SCSS  @each $name in list, detail {  .jdc\_#{$name} {  background-image: url(/img/jdc\_#{$name}.png);  }  }  // CSS  .jdc\_list {  background-image: url(/img/jdc\_list.png);  background-color: red;  }  .jdc\_detail {  background-image: url(/img/jdc\_detail.png);  background-color: blue;  }  // SCSS  @each $name, $color in (list, red), (detail, blue) {  .jdc\_#{$name} {  background-image: url(/img/jdc\_#{$name}.png);  background-color: $color;  }  } |

## function 函数

|  |
| --- |
| @function pxToRem($px) {  @return $px / 10px \* 1rem;  }  .jdc {  font-size: pxToRem(12px);  } |

## 运算规范

运算符之间空出一个空格

|  |
| --- |
| .jdc {  width: 100px - 50px;  height: 30px / 5;  } |

注意运算单位，单位同时参与运算，所以 10px 不等于 10，乘除运算时需要特别注意

|  |
| --- |
| // 正确的运算格式  .jdc {  width: 100px - 50px;  width: 100px + 50px;  width: 100px \* 2;  width: 100px / 2;  } |

## 重置样式

可使用normalize.css 或以下css

## 移动端

|  |
| --- |
| \* { -webkit-tap-highlight-color: transparent; outline: 0; margin: 0; padding: 0; vertical-align: baseline; }  body, h1, h2, h3, h4, h5, h6, hr, p, blockquote, dl, dt, dd, ul, ol, li, pre, form, fieldset, legend, button, input, textarea, th, td { margin: 0; padding: 0; vertical-align: baseline; }  img { border: 0 none; vertical-align: top; }  i, em { font-style: normal; }  ol, ul { list-style: none; }  input, select, button, h1, h2, h3, h4, h5, h6 { font-size: 100%; font-family: inherit; }  table { border-collapse: collapse; border-spacing: 0; }  a { text-decoration: none; color: #666; }  body { margin: 0 auto; min-width: 320px; max-width: 640px; height: 100%; font-size: 14px; fon-family: -apple-system,Helvetica,sans-serif; line-height: 1.5; color: #666; -webkit-text-size-adjust: 100% !important; text-size-adjust: 100% !important; }  input[type="text"], textarea { -webkit-appearance: none; -moz-appearance: none; appearance: none; } |

## PC端

|  |
| --- |
| html, body, div, h1, h2, h3, h4, h5, h6, p, dl, dt, dd, ol, ul, li, fieldset, form, label, input, legend, table, caption, tbody, tfoot, thead, tr, th, td, textarea, article, aside, audio, canvas, figure, footer, header, mark, menu, nav, section, time, video { margin: 0; padding: 0; }  h1, h2, h3, h4, h5, h6 { font-size: 100%; font-weight: normal }  article, aside, dialog, figure, footer, header, hgroup, nav, section, blockquote { display: block; }  ul, ol { list-style: none; }  img { border: 0 none; vertical-align: top; }  blockquote, q { quotes: none; }  blockquote:before, blockquote:after, q:before, q:after { content: none; }  table { border-collapse: collapse; border-spacing: 0; }  strong, em, i { font-style: normal; font-weight: normal; }  ins { text-decoration: underline; }  del { text-decoration: line-through; }  mark { background: none; }  input::-ms-clear { display: none !important; }  body { font: 12px/1.5 \5FAE\8F6F\96C5\9ED1, \5B8B\4F53, "Hiragino Sans GB", STHeiti, "WenQuanYi Micro Hei", "Droid Sans Fallback", SimSun, sans-serif; background: #fff; }  a { text-decoration: none; color: #333; }  a:hover { text-decoration: underline; } |

# 媒体查询

设备尺寸参考 ：[Mobile devices](http://mydevice.io/devices/)

媒体查询类型浏览器支持情况：[CSS3 Media Queries overview](http://cssmediaqueries.com/overview.html)

## 常用查询语句

判断设备横竖屏

|  |
| --- |
| /\* 横屏 \*/  @media all and (orientation :landscape) {  }  /\* 竖屏 \*/  @media all and (orientation :portrait) {  } |

判断设备宽高

|  |
| --- |
| /\* 设备宽度大于 320px 小于 640px \*/  @media all and (min-width:320px) and (max-width:640px) {    } |

判断设备像素比

|  |
| --- |
| /\* 设备像素比为 1 \*/  @media only screen and (-webkit-min-device-pixel-ratio: 1), only screen and (min-device-pixel-ratio: 1) {    }  /\* 设备像素比为 1.5 \*/  @media only screen and (-webkit-min-device-pixel-ratio: 1.5), only screen and (min-device-pixel-ratio: 1.5) {    }  /\* 设备像素比为 2 \*/  @media only screen and (-webkit-min-device-pixel-ratio: 2), only screen and (min-device-pixel-ratio: 2) {    } |

## 常用设备设置

### iPhones

|  |
| --- |
| /\* ----------- iPhone 4 and 4S ----------- \*/  /\* Portrait and Landscape \*/  @media only screen  and (min-device-width: 320px)  and (max-device-width: 480px)  and (-webkit-min-device-pixel-ratio: 2) {  }  /\* Portrait \*/  @media only screen  and (min-device-width: 320px)  and (max-device-width: 480px)  and (-webkit-min-device-pixel-ratio: 2)  and (orientation: portrait) {  }  /\* Landscape \*/  @media only screen  and (min-device-width: 320px)  and (max-device-width: 480px)  and (-webkit-min-device-pixel-ratio: 2)  and (orientation: landscape) {  }  /\* ----------- iPhone 5 and 5S ----------- \*/  /\* Portrait and Landscape \*/  @media only screen  and (min-device-width: 320px)  and (max-device-width: 568px)  and (-webkit-min-device-pixel-ratio: 2) {  }  /\* Portrait \*/  @media only screen  and (min-device-width: 320px)  and (max-device-width: 568px)  and (-webkit-min-device-pixel-ratio: 2)  and (orientation: portrait) {  }  /\* Landscape \*/  @media only screen  and (min-device-width: 320px)  and (max-device-width: 568px)  and (-webkit-min-device-pixel-ratio: 2)  and (orientation: landscape) {  }  /\* ----------- iPhone 6 ----------- \*/  /\* Portrait and Landscape \*/  @media only screen  and (min-device-width: 375px)  and (max-device-width: 667px)  and (-webkit-min-device-pixel-ratio: 2) {  }  /\* Portrait \*/  @media only screen  and (min-device-width: 375px)  and (max-device-width: 667px)  and (-webkit-min-device-pixel-ratio: 2)  and (orientation: portrait) {  }  /\* Landscape \*/  @media only screen  and (min-device-width: 375px)  and (max-device-width: 667px)  and (-webkit-min-device-pixel-ratio: 2)  and (orientation: landscape) {  }  /\* ----------- iPhone 6+ ----------- \*/  /\* Portrait and Landscape \*/  @media only screen  and (min-device-width: 414px)  and (max-device-width: 736px)  and (-webkit-min-device-pixel-ratio: 3) {  }  /\* Portrait \*/  @media only screen  and (min-device-width: 414px)  and (max-device-width: 736px)  and (-webkit-min-device-pixel-ratio: 3)  and (orientation: portrait) {  }  /\* Landscape \*/  @media only screen  and (min-device-width: 414px)  and (max-device-width: 736px)  and (-webkit-min-device-pixel-ratio: 3)  and (orientation: landscape) {  } |

### Galaxy Phones

|  |
| --- |
| /\* ----------- Galaxy S3 ----------- \*/  /\* Portrait and Landscape \*/  @media screen  and (device-width: 320px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 2) {  }  /\* Portrait \*/  @media screen  and (device-width: 320px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 2)  and (orientation: portrait) {  }  /\* Landscape \*/  @media screen  and (device-width: 320px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 2)  and (orientation: landscape) {  }  /\* ----------- Galaxy S4 ----------- \*/  /\* Portrait and Landscape \*/  @media screen  and (device-width: 320px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 3) {  }  /\* Portrait \*/  @media screen  and (device-width: 320px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 3)  and (orientation: portrait) {  }  /\* Landscape \*/  @media screen  and (device-width: 320px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 3)  and (orientation: landscape) {  }  /\* ----------- Galaxy S5 ----------- \*/  /\* Portrait and Landscape \*/  @media screen  and (device-width: 360px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 3) {  }  /\* Portrait \*/  @media screen  and (device-width: 360px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 3)  and (orientation: portrait) {  }  /\* Landscape \*/  @media screen  and (device-width: 360px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 3)  and (orientation: landscape) {  } |

### HTC Phones

|  |
| --- |
| /\* ----------- HTC One ----------- \*/  /\* Portrait and Landscape \*/  @media screen  and (device-width: 360px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 3) {  }  /\* Portrait \*/  @media screen  and (device-width: 360px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 3)  and (orientation: portrait) {  }  /\* Landscape \*/  @media screen  and (device-width: 360px)  and (device-height: 640px)  and (-webkit-device-pixel-ratio: 3)  and (orientation: landscape) {  } |

### iPads

|  |
| --- |
| /\* ----------- iPad mini ----------- \*/  /\* Portrait and Landscape \*/  @media only screen  and (min-device-width: 768px)  and (max-device-width: 1024px)  and (-webkit-min-device-pixel-ratio: 1) {  }  /\* Portrait \*/  @media only screen  and (min-device-width: 768px)  and (max-device-width: 1024px)  and (orientation: portrait)  and (-webkit-min-device-pixel-ratio: 1) {  }  /\* Landscape \*/  @media only screen  and (min-device-width: 768px)  and (max-device-width: 1024px)  and (orientation: landscape)  and (-webkit-min-device-pixel-ratio: 1) {  }  /\* ----------- iPad 1 and 2 ----------- \*/  /\* Portrait and Landscape \*/  @media only screen  and (min-device-width: 768px)  and (max-device-width: 1024px)  and (-webkit-min-device-pixel-ratio: 1) {  }  /\* Portrait \*/  @media only screen  and (min-device-width: 768px)  and (max-device-width: 1024px)  and (orientation: portrait)  and (-webkit-min-device-pixel-ratio: 1) {  }  /\* Landscape \*/  @media only screen  and (min-device-width: 768px)  and (max-device-width: 1024px)  and (orientation: landscape)  and (-webkit-min-device-pixel-ratio: 1) {  }  /\* ----------- iPad 3 and 4 ----------- \*/  /\* Portrait and Landscape \*/  @media only screen  and (min-device-width: 768px)  and (max-device-width: 1024px)  and (-webkit-min-device-pixel-ratio: 2) {  }  /\* Portrait \*/  @media only screen  and (min-device-width: 768px)  and (max-device-width: 1024px)  and (orientation: portrait)  and (-webkit-min-device-pixel-ratio: 2) {  }  /\* Landscape \*/  @media only screen  and (min-device-width: 768px)  and (max-device-width: 1024px)  and (orientation: landscape)  and (-webkit-min-device-pixel-ratio: 2) {  } |

# JS规范

## 语言规范

JavaScript 是一种客户端脚本语言，这里列出了编写 JavaScript 时需要遵守的规则。

**类型**

* 基本类型
  + 字符串
  + 数值
  + 布尔类型
  + null
  + undefined

|  |
| --- |
| const foo = 1  let bar = foo  bar = 9  console.log(foo, bar) // 1, 9 |

* 复杂类型
  + object
  + array
  + function

|  |
| --- |
| const foo = [1, 2, 3]  const bar = foo  bar[0] = 9  console.log(foo[0], bar[0]) // 9, 9 |

**引用**

const 和 let 都是块级作用域，var 是函数级作用域

* 对所有引用都使用 const，不要使用 var

|  |
| --- |
| // bad  var a = 1  var b = 2  // good  const a = 1  const b = 2 |

* 如果引用是可变动的，则使用 let

|  |
| --- |
| // bad  var count = 1  if (count < 10) {  count += 1  }  // good  let count = 1  if (count < 10) {  count += 1  } |

**对象**

* 请使用字面量值创建对象

|  |
| --- |
| // bad  const a = new Object{}  // good  const a = {} |

* 别使用保留字作为对象的键值，这样在 IE8 下不会运行

|  |
| --- |
| // bad  const a = {  default: {}, // default 是保留字  common: {}  }  // good  const a = {  defaults: {},  common: {}  } |

* 请使用对象方法的简写方式

|  |
| --- |
| // bad  const item = {  value: 1,  addValue: function (val) {  return item.value + val  }  }  // good  const item = {  value: 1,  addValue(val) {  return item.value + val  }  } |

* 请使用对象属性值的简写方式

|  |
| --- |
| const job = 'FrontEnd'  // bad  const item = {  job: job  }  // good  const item = {  job  } |

* 对象属性值的简写方式要和声明式的方式分组

|  |
| --- |
| const job = 'FrontEnd'  const department = 'JDC'  // bad  const item = {  sex: 'male',  job,  age: 25,  department  }  // good  const item = {  job,  department,  sex: 'male',  age: 25  } |

**数组**

* 请使用字面量值创建数组

|  |
| --- |
| // bad  const items = new Array()  // good  const items = [] |

* 向数组中添加元素时，请使用 push 方法

|  |
| --- |
| const items = []  // bad  items[items.length] = 'test'  // good  items.push('test') |

* 使用拓展运算符 ... 复制数组

|  |
| --- |
| // bad  const items = []  const itemsCopy = []  const len = items.length  let i  // bad  for (i = 0; i < len; i++) {  itemsCopy[i] = items[i]  }  // good  itemsCopy = [...items] |

* 使用数组的 map 等方法时，请使用 return 声明，如果是单一声明语句的情况，可省略 return

|  |
| --- |
| // good  [1, 2, 3].map(x => {  const y = x + 1  return x \* y  })  // good  [1, 2, 3].map(x => x + 1)  // bad  const flat = {}  [[0, 1], [2, 3], [4, 5]].reduce((memo, item, index) => {  const flatten = memo.concat(item)  flat[index] = flatten  })  // good  const flat = {}  [[0, 1], [2, 3], [4, 5]].reduce((memo, item, index) => {  const flatten = memo.concat(item)  flat[index] = flatten  return flatten  })  // bad  inbox.filter((msg) => {  const { subject, author } = msg  if (subject === 'Mockingbird') {  return author === 'Harper Lee'  } else {  return false  }  })  // good  inbox.filter((msg) => {  const { subject, author } = msg  if (subject === 'Mockingbird') {  return author === 'Harper Lee'  }  return false  }) |

**解构赋值**

* 当需要使用对象的多个属性时，请使用解构赋值

|  |
| --- |
| // bad  function getFullName (user) {  const firstName = user.firstName  const lastName = user.lastName  return `${firstName} ${lastName}`  }  // good  function getFullName (user) {  const { firstName, lastName } = user  return `${firstName} ${lastName}`  }  // better  function getFullName ({ firstName, lastName }) {  return `${firstName} ${lastName}`  } |

* 当需要使用数组的多个值时，请同样使用解构赋值

|  |
| --- |
| const arr = [1, 2, 3, 4]  // bad  const first = arr[0]  const second = arr[1]  // good  const [first, second] = arr |

* 函数需要回传多个值时，请使用对象的解构，而不是数组的解构

|  |
| --- |
| // bad  function doSomething () {  return [top, right, bottom, left]  }  // 如果是数组解构，那么在调用时就需要考虑数据的顺序  const [top, xx, xxx, left] = doSomething()  // good  function doSomething () {  return { top, right, bottom, left }  }  // 此时不需要考虑数据的顺序  const { top, left } = doSomething() |

**字符串**

* 字符串统一使用单引号的形式 ''

|  |
| --- |
| // bad  const department = "JDC"  // good  const department = 'JDC' |

* 字符串太长的时候，请不要使用字符串连接符换行 \，而是使用 +

|  |
| --- |
| const str = '凹凸实验室 凹凸实验室 凹凸实验室' +  '凹凸实验室 凹凸实验室 凹凸实验室' +  '凹凸实验室 凹凸实验室' |

* 程序化生成字符串时，请使用模板字符串

|  |
| --- |
| const test = 'test'  // bad  const str = ['a', 'b', test].join()  // bad  const str = 'a' + 'b' + test  // good  const str = `ab${test}` |

**函数**

* 请使用函数声明，而不是函数表达式

|  |
| --- |
| // bad  const foo = function () {  // do something  }  // good  function foo () {  // do something  } |

* 不要在非函数代码块中声明函数

|  |
| --- |
| // bad  if (isUse) {  function test () {  // do something  }  }  // good  let test  if (isUse) {  test = () => {  // do something  }  } |

* 不要使用 arguments，可以选择使用 ...

arguments 只是一个类数组，而 ... 是一个真正的数组

|  |
| --- |
| // bad  function test () {  const args = Array.prototype.slice.call(arguments)  return args.join('')  }  // good  function test (...args) {  return args.join('')  } |

* 不要更改函数参数的值

|  |
| --- |
| // bad  function test (opts) {  opts = opts || {}  }  // good  function test (opts = {}) {  // ...  } |

**原型**

* 使用 class，避免直接操作 prototype

|  |
| --- |
| // bad  function Queue (contents = []) {  this.\_queue = [..contents]  }  Queue.prototype.pop = function () {  const value = this.\_queue[0]  this.\_queue.splice(0, 1)  return value  }  // good  class Queue {  constructor (contents = []) {  this.\_queue = [...contents]  }  pop () {  const value = this.\_queue[0]  this.\_queue.splice(0, 1)  return value  }  } |

**模块**

* 使用标准的 ES6 模块语法 import 和 export

|  |
| --- |
| // bad  const util = require('./util')  module.exports = util  // good  import Util from './util'  export default Util  // better  import { Util } from './util'  export default Util |

* 不要使用 import 的通配符 \*，这样可以确保你只有一个默认的 export

|  |
| --- |
| // bad  import \* as Util from './util'  // good  import Util from './util' |

**迭代器**

* 不要使用 iterators

|  |
| --- |
| const numbers = [1, 2, 3, 4, 5]  // bad  let sum = 0  for (let num of numbers) {  sum += num  }  // good  let sum = 0  numbers.forEach(num => sum += num)  // better  const sum = numbers.reduce((total, num) => total + num, 0) |

**对象属性**

* 使用 . 来访问对象属性

|  |
| --- |
| const joke = {  name: 'haha',  age: 28  }  // bad  const name = joke['name']  // good  const name = joke.name |

**变量声明**

* 声明变量时，请使用 const、let 关键字，如果没有写关键字，变量就会暴露在全局上下文中，这样很可能会和现有变量冲突，另外，也很难明确该变量的作用域是什么。这里推荐使用 const 来声明变量，我们需要避免全局命名空间的污染。

|  |
| --- |
| // bad  demo = new Demo()  // good  const demo = new Demo() |

* 将所有的 const 和 let 分组

|  |
| --- |
| // bad  let a  const b  let c  const d  let e  // good  const b  const d  let a  let c  let e |

**Hoisting**

* var 存在变量提升的情况，即 var 声明会被提升至该作用域的顶部，但是他们的赋值并不会。而 const 和 let 并不存在这种情况，他们被赋予了 [Temporal Dead Zones, TDZ](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let#Temporal_dead_zone_and_errors_with_let)

|  |
| --- |
| function example () {  console.log(notDefined) // => throws a ReferenceError  }  function example () {  console.log(declareButNotAssigned) // => undefined  var declaredButNotAssigned = true  }  function example () {  let declaredButNotAssigned  console.log(declaredButNotAssigned) // => undefined  declaredButNotAssigned = true  }  function example () {  console.log(declaredButNotAssigned) // => throws a ReferenceError  console.log(typeof declaredButNotAssigned) // => throws a ReferenceError  const declaredButNotAssigned = true  } |

* 匿名函数的变量名会提升，但函数内容不会

|  |
| --- |
| function example () {  console.log(anonymous) // => undefined  anonymous()  var anonymous = function () {  console.log('test')  }  } |

* 命名的函数表达式的变量名会被提升，但函数名和函数函数内容并不会

|  |
| --- |
| function example() {  console.log(named) // => undefined  named() // => TypeError named is not a function  superPower() // => ReferenceError superPower is not defined  var named = function superPower () {  console.log('Flying')  }  }  function example() {  console.log(named) // => undefined  named() // => TypeError named is not a function  var named = function named () {  console.log('named')  }  } |

**分号**

* 我们遵循 Standard 的规范，不使用分号。

关于应不应该使用分号的讨论有很多，本规范认为非必要的时候，应该不使用分号，好的 JS 程序员应该清楚场景下是一定要加分号的，相信你也是名好的开发者。

|  |
| --- |
| // bad  const test = 'good';  (function () {  const str = 'hahaha';  })()  // good  const test = 'good'  ;(() => {  const str = 'hahaha'  })(); |

**标准特性**

为了代码的可移植性和兼容性，我们应该最大化的使用标准方法，例如优先使用 string.charAt(3) 而不是 string[3]

**eval()**

由于 eval 方法比较 evil，所以我们约定禁止使用该方法

**with() {}**

由于 with 方法会产生神奇的作用域，所以我们也是禁止使用该方法的

**for-in 循环**

推荐使用 for in 语法，但是在对对象进行操作时，容易忘了检测 hasOwnProperty(key)，所以我们启用了 ESLint 的 guard-for-in 选项

对数组进行 for in 的时候，顺序是不固定的

**修改内置对象的原型**

不要修改内置对象，如 Object 和 Array

## 代码规范

## 编码规范

统一团队的编码规范，有助于代码的维护。本章是传统意义上的 Style Guideline，目的是统一一些相对主观化的代码风格。

### 单行代码块

在单行代码块中使用空格

不推荐

|  |
| --- |
| function foo () {return true}  if (foo) {bar = 0} |

推荐

|  |
| --- |
| function foo () { return true }  if (foo) { bar = 0 } |

### 大括号风格

在编程过程中，大括号风格与缩进风格紧密联系，用来描述大括号相对代码块位置的方法有很多。在 JavaScript 中，主要有三种风格，如下：

* **One True Brace Style**

|  |
| --- |
| if (foo) {  bar()  } else {  baz()  } |

* **Stroustrup**

|  |
| --- |
| if (foo) {  bar()  }  else {  baz()  } |

* **Allman**

|  |
| --- |
| if (foo)  {  bar()  }  else  {  baz()  } |

我们团队约定使用 One True Brace Style 风格

### 变量命名

当命名变量时，主流分为驼峰式命名（variableName）和下划线命名（variable\_name）两大阵营。

团队约定使用驼峰式命名

### 拖尾逗号

在 ECMAScript5 里面，对象字面量中的拖尾逗号是合法的，但在 IE8（非 IE8 文档模式）下，当出现拖尾逗号，则会抛出错误。

拖尾逗号的例子：

|  |
| --- |
| var foo = {  name: 'foo',  age: '22',  } |

拖尾逗号的好处是，简化了对象和数组添加或删除元素，我们只需要修改新增的行即可，并不会增加差异化的代码行数。

因为拖尾逗号有好也有不好，所以团队约定允许在最后一个元素或属性与闭括号 ] 或 } 在不同行时，可以（但不要求）使用拖尾逗号。当在同一行时，禁止使用拖尾逗号。

### 逗号空格

逗号前后的空格可以提高代码的可读性，团队约定在逗号后面使用空格，逗号前面不加空格。

不推荐

|  |
| --- |
| var foo = 1,bar = 2  var foo = 1 , bar = 2  var foo = 1 ,bar = 2 |

推荐

|  |
| --- |
| var foo = 1, bar = 2 |

### 逗号风格

逗号分隔列表时，在 JavaScript 中主要有两种逗号风格：

* 标准风格，逗号放置在当前行的末尾
* 逗号前置风格，逗号放置在下一行的开始位置

团队约定使用标准风格

不推荐

|  |
| --- |
| var foo = 1  ,  bar = 2  var foo = 1  , bar = 2  var foo = ['name'  , 'age'] |

推荐

|  |
| --- |
| var foo = 1,  bar = 2  var foo = ['name',  'age'] |

### 计算属性的空格

团队约定在对象的计算属性内，禁止使用空格

不推荐

|  |
| --- |
| obj['foo' ]  obj[ 'foo']  obj[ 'foo' ] |

推荐

|  |
| --- |
| obj['foo'] |

### 拖尾换行

在非空文件中，存在拖尾换行是一个常见的 UNIX 风格，它的好处是可以方便在串联和追加文件时不会打断 Shell 的提示。在日常的项目中，保留拖尾换行的好处是，可以减少版本控制时的代码冲突。

不推荐

|  |
| --- |
| function func () {  // do something  } |

推荐

|  |
| --- |
| function func () {  // do something  }  // 此处是新的一行 |

可以通过 .editorconfig 添加 EOL

### 函数调用

为了避免语法错误，团队约定在函数调用时，禁止使用空格

不推荐

|  |
| --- |
| fn ()  fn  () |

推荐

|  |
| --- |
| fn() |

### 缩进

代码保持一致的缩进，是作为工程师的职业素养。但缩进用两个空格，还是四个空格，是用 Tab 还是空格呢？这样的争论太多了，也得不出答案。本规范结合了市面上优秀的开源项目，姑且约定使用 空格 来缩进，而且缩进使用两个空格。

那是不是不能使用 Tab 进行缩进了？我们可以通过配置 .editorconfig ，将 Tab 自动转换为空格。

### 对象字面量的键值缩进

团队约定对象字面量的键和值之间不能存在空格，且要求对象字面量的冒号和值之间存在一个空格

不推荐

|  |
| --- |
| var obj = { 'foo' : 'haha' } |

推荐

|  |
| --- |
| var obj = { 'foo': 'haha' } |

### 构造函数首字母大写

在 JavaScript 中 new 操作符用来创建某个特定类型的对象的一个实例，该类型的对象是由一个构造函数表示的。由于构造函数只是常规函数，唯一区别是使用 new 来调用。所以我们团队约定构造函数的首字母要大小，以此来区分构造函数和普通函数。

不推荐

|  |
| --- |
| var fooItem = new foo() |

推荐

|  |
| --- |
| var fooItem = new Foo() |

### 构造函数的参数

在 JavaScript 中，通过 new 调用构造函数时，如果不带参数，可以省略后面的圆括号。但这样会造成与整体的代码风格不一致，所以团队约定使用圆括号

不推荐

|  |
| --- |
| var person = new Person |

推荐

|  |
| --- |
| var person = new Person() |

### 链式调用

链式调用如果放在同一行，往往会造成代码的可读性差，但有些时候，短的链式调用并不会影响美观。所以本规范约定一行最多只能有四个链式调用，超过就要求换行。

### 空行

空白行对于分离代码逻辑有帮助，但过多的空行会占据屏幕的空间，影响可读性。团队约定最大连续空行数为 2

不推荐

|  |
| --- |
| var a = 1  var b = 2 |

推荐

|  |
| --- |
| var a = 1  var b = 2 |

### 链式赋值

链式赋值容易造成代码的可读性差，所以团队约定禁止使用链式赋值

不推荐

|  |
| --- |
| var a = b = c = 1 |

推荐

|  |
| --- |
| var a = 1  var b = 1  var c = 1 |

### 变量声明

JavaScript 允许在一个声明中，声明多个变量。团队约定在声明变量时，一个声明只能有一个变量

不推荐

|  |
| --- |
| var a, b, c |

推荐

|  |
| --- |
| var a  var b  var c |

### 分号

JavaScript 在所有类 C 语言中是比较独特的，它不需要在每个语句的末尾有分号。在很多情况下，JavaScript 引擎可以确定一个分号应该在什么位置然后自动添加它。此特征被称为 自动分号插入 (ASI)，被认为是 JavaScript 中较为有争议的特征。

团队中对于是否应该使用分号，也有许多争论，本规范推荐不使用分号，因为我们认为好的工程师应该知道什么时候该加，什么时候不该加。

相关参考 ：[semi](http://eslint.org/docs/rules/semi)

### 代码块空格

一致性是任何风格指南的重要组成部分。虽然在哪里放置块的开括号纯属个人偏好，但在整个项目中应该保持一致。不一致的风格将会分散读者阅读代码的注意力。

团队约定代码块前要添加空格

不推荐

|  |
| --- |
| if (a){  b()  }  function a (){} |

推荐

|  |
| --- |
| if (a) {  b()  }  function a () {} |

### 函数声明的空格

当格式化一个函数，函数名或 function 关键字与左括号之间允许有空白。命名函数要求函数名和 function 关键字之间有空格，但是匿名函数要求不加空格。

团队约定函数括号前要加空格

不推荐

|  |
| --- |
| function func(x) {  // ...  } |

推荐

|  |
| --- |
| function func (x) {  // ...  } |

### 操作符的空格

团队约定操作符前后都需要添加空格

不推荐

|  |
| --- |
| var sum = 1+2 |

推荐

|  |
| --- |
| var sum = 1 + 2 |

### BOM

Unicode 字节顺序标记 (BOM) 用来指定代码单元是高字节序还是低字节序。也就是说，是高位在前还是低位在前。UTF-8 不需要 BOM 来表明字节顺序，因为单个字节并不影响字节顺序。

相信不少同学遇到过 BOM 的坑，这里不多说了，切记不要使用 windows 的记事本改代码！