

目录

介绍	1.1
HTML 规范	1.2
代码规范	1.2.1
注释规范	1.2.2
文件模版	1.2.3
WebApp Meta	1.2.4
图片规范	1.3
图片格式	1.3.1
图片大小	1.3.2
图片质量	1.3.3
图片引入	1.3.4
CSS 规范	1.4
代码规范	1.4.1
注释规范	1.4.2
SASS 规范	1.4.3
重置样式	1.4.4
媒体查询	1.4.5
移动端常用私有属性	1.4.6
命名规范	1.5
目录命名	1.5.1
图片命名	1.5.2
HTML/CSS 命名	1.5.3
ClassName 命名	1.5.4
JS 规范	1.6
语言规范	1.6.1
代码规范	1.6.2

- [1.1.1. 111](#)

1.1.1. 111

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间： 2018-11-07
18:44:39

- **1.2.1. 前端开发编码规范**
 - **1.2.1.1. HTML规范**
 - **1.2.1.1.1. 代码规范**
 - **1.2.1.1.2. 注释规范**
 - **1.2.1.1.3. WebApp Meta 标签设置(iOS)**
 - **1.2.1.1.4. HTML模版**

1.2.1. 前端开发编码规范

基于 [W3C](#)、[苹果开发者](#) 等官方文档，并结合团队日常业务需求以及团队在日常开发过程中总结提炼出的经验而约定。

旨在增强团队开发协作、提高代码质量和打造开发基石的编码规范，是团队代码基本约定的内容，必须严格遵循。

1.2.1.1. HTML规范

1.2.1.1.1. 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.
-
1. 一个ASCII字符串 "<!DOCTYPE"，大小写不敏感
 2. 一个或多个空白字符
 3. 一个ASCII字符串"html"，大小写不敏感
 4. 一个可选的历史遗留的DOCTYPE字符串（[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](#)）字符串
 5. 一个或多个空白字符
 6. 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

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](#)

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:

<code>zh-SG</code> 中文 (简体, 新加坡)	对应 <code>cmn-Hans-SG</code> 普通话 (简体, 新加坡)
<code>zh-HK</code> 中文 (繁体, 香港)	对应 <code>cmn-Hant-HK</code> 普通话 (繁体, 香港)
<code>zh-MO</code> 中文 (繁体, 澳门)	对应 <code>cmn-Hant-MO</code> 普通话 (繁体, 澳门)
<code>zh-TW</code> 中文 (繁体, 台湾)	对应 <code>cmn-Hant-TW</code> 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](#) 里面的 “Type: redundant”

更多关于 Languages Tags :

[W3C Language tags in HTML and XML](#)

[网页头部的声明应该是用 lang="zh" 还是 lang="zh-cn"?](#)

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](#)

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](#)，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?](#)

[GBK: Application of IANA Charset Registration for GBK](#)

[Charset : 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](#)，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结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](#)

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](#)

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](#)

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>
```

1.2.1.1.2. 注释规范

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/#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 -->
```

1.2.1.1.3. WebApp Meta 标签设置(iOS)

A web application is designed to look and behave in a way similar to a native application—for example, it is scaled to fit the entire screen on iOS. You can tailor your web application for Safari on iOS even further, by making it appear like a native application when the user adds it to the Home screen. You do this by using settings for iOS that are ignored by other platforms.

WebApp目的在于使其界面和行为在某种程度上类似于原生APP应用。例如，WebApp 可以在 iOS 设备上通过缩放去适配设备屏幕。当用户将WebApp程序添加到主屏幕后，会使得它看上去像原生APP一样，以此，你可以进一步为 Safari 定制自己的 WebApp，而使用某些专为 iOS 平台设定的设置就可以做到。

WebApp可以通过设置 meta 标签来改变页面的一些表现，有些 meta 设置在 Safari 应用或原生 App 的内嵌网页中都可以生效，而有些设置则需要将应用添加到主屏幕的时候才会生效。

Viewport Meta Tag

通用类设置：

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
```

- width -- viewport的宽度
- height -- viewport的高度
- initial-scale -- 初始的缩放比例
- minimum-scale -- 允许用户缩放到的最小比例

- `maximum-scale` -- 允许用户缩放到的最大比例
- `user-scalable` -- 是否允许用户缩放

其中 Safari on iOS viewport:

The width of the viewport in pixels. The default is 980. The range is from 200 to 10,000.

The `minimum-scale` and `maximum-scale` properties also affect the behavior when changing orientations. The range of these property values is from >0 to 10.0. The default value for `minimum-scale` is 0.25 and `maximum-scale` is 5.0

`user-scalable` -- The default is yes. Setting `user-scalable` to no also prevents a webpage from scrolling when entering text in an input field.

- 默认宽度是 980px, 范围从 200px 到 10000px
- `initial-scale` 缩放比例范围值是从 >0 到 10 之间
- `minimum-scale` 默认值是 0.25
- `maximum-scale` 默认值是 5
- `user-scalable` -- 默认值是 yes, 设置 no 还可以在文本框输入文本的时候阻止页面滚动

更多关于 Safari on iOS viewport 的设置:

[Configuring the Viewport](#)

[Safari HTML Reference](#)

Apple-Specific Meta Tag Keys

apple-mobile-web-app-capable

设置 WebApp 是否进入全屏模式，该设置需要添加到主屏幕才生效

```
<meta name="apple-mobile-web-app-capable" content="yes">
```

If `content` is set to yes, the web application runs in full-screen mode; otherwise, it does not. The default behavior is to use Safari to display web content. You can determine whether a webpage is displayed in full-screen mode using the `window.navigator.standalone` read-only Boolean JavaScript property.

- `content`设置 yes 进入全屏模式
- 默认会启动 Safari 应用，使用 Safari 应用浏览
- 通过检测 `window.navigator.standalone` 的 Boolean 值可以判断 web 应用是否处于全屏模式

apple-mobile-web-app-status-bar-style

为 webapp 设置状态栏样式

```
<meta name="apple-mobile-web-app-status-bar-style" content="black">
```

This meta tag has no effect unless you first specify full-screen mode as described in `apple-mobile-web-app-capable`.

If content is set to default, the status bar appears normal. If set to black, the status bar has a black background. If set to black-translucent, the status bar is black and translucent. If set to default or black, the web content is displayed below the status bar. If set to black-translucent, the web content is displayed on the entire screen, partially obscured by the status bar. The default value is default.

- 此 meta 设置只在全屏模式生效
- 默认值是 default
- `content="black"`, 状态栏背景黑色, 网页内容在状态栏下面
- `content="black-translucent"`, 状态栏半透明, 背景黑色, 网页内容占满全屏

该设置在 iOS6 和 iOS7 表现还可以, 但到了 iOS8 后会出现各种问题, 而且在 iOS9 中并没有生效。

参阅: [iOS 8: web app status bar position and resizing problems](#)

format-detection

自动识别页面中有可能是电话格式的数字

```
<meta name="format-detection" content="telephone=no">
```

By default, Safari on iOS detects any string formatted like a phone number and makes it a link that calls the number. Specifying `telephone=no` disables this feature.

iOS中的 Safari 会默认识别与电话格式相似的数字并生成一个可以拉起电话应用并将该数字作为电话号码拨打的链接。定义 `telephone=no` 可以屏蔽该功能

更多 WebApp 设置参考 [Configuring Web Applications](#)

1.2.1.1.4. HTML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 `meta` 标签的设置, 以下是 PC 和移动端的 HTML 模版

HTML5最简结构标准模版:

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
<meta charset="UTF-8">
<title>HTML5最简结构标准模版</title>
</head>
<body>

</body>
</html>
```

团队约定的HTML模版

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" >
<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>
```

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间： 2018-06-04
02:34:32

title: 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.
-
1. 一个ASCII字符串 "<!DOCTYPE"，大小写不敏感
 2. 一个或多个空白字符
 3. 一个ASCII字符串"html"，大小写不敏感
 4. 一个可选的历史遗留的DOCTYPE字符串（[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](#)）字符串
 5. 一个或多个空白字符
 6. 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](#)

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](#)

团队约定

推荐使用属性值 `cmm-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](#) 里面的 “Type: redundant”

更多关于 Languages Tags :

[W3C Language tags in HTML and XML](#)

[网页头部的声明应该是用 lang="zh" 还是 lang="zh-cn"?](#)

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](#)

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](#)，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?](#)

[GBK: Application of IANA Charset Registration for GBK](#)

[Charset : 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，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结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](#)

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](#)

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](#)

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>
```

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间： 2018-06-04
02:34:32

title: 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/#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 -->
```

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间： 2018-06-04
02:34:32

title: 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 meta 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>
```

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间： 2018-06-04
02:34:32

title: WebApp Meta

WebApp Meta 标签设置(iOS)

A web application is designed to look and behave in a way similar to a native application—for example, it is scaled to fit the entire screen on iOS. You can tailor your web application for Safari on iOS even further, by making it appear like a native application when the user adds it to the Home screen. You do this by using settings for iOS that are ignored by other platforms.

WebApp目的在于使其界面和行为在某种程度上类似于原生APP应用。例如，WebApp可以在iOS设备上通过缩放去适配设备屏幕。当用户将WebApp程序添加到主屏幕后，会使得它看上去像原生APP一样，以此，你可以进一步为Safari定制自己的WebApp，而使用某些专为iOS平台设定的设置就可以做到。

WebApp可以通过设置meta标签来改变页面的一些表现，有些meta设置在Safari应用或原生App的内嵌网页中都可以生效，而有些设置则需要将应用添加到主屏幕的时候才会生效。

Viewport Meta Tag

通用类设置

```
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">
```

- width -- viewport的宽度
- height -- viewport的高度
- initial-scale -- 初始的缩放比例
- minimum-scale -- 允许用户缩放到的最小比例
- maximum-scale -- 允许用户缩放到的最大比例
- user-scalable -- 是否允许用户缩放

Safari on iOS viewport

The width of the viewport in pixels. The default is 980. The range is from 200 to 10,000.

The minimum-scale and maximum-scale properties also affect the behavior when changing orientations. The range of these property values is from >0 to 10.0. The default value for minimum-scale is 0.25 and maximum-scale is 5.0

user-scalable -- The default is yes. Setting user-scalable to no also prevents a webpage from scrolling when entering text in an input field.

- 默认宽度是980px，范围从200px到10000px
- initial-scale缩放比例范围值是从>0到10之间
- minimum-scale默认值是0.25

- maximum-scale 默认值是 5
- user-scalable -- 默认值是 yes, 设置 no 还可以在文本框输入文本的时候阻止页面滚动

更多关于 Safari on iOS viewport 的设置:

[Configuring the Viewport Safari HTML Reference](#)

Apple-Specific Meta Tag Keys

apple-mobile-web-app-capable

设置 WebApp 是否进入全屏模式，该设置需要添加到主屏幕才生效

```
<meta name="apple-mobile-web-app-capable" content="yes">
```

If content is set to yes, the web application runs in full-screen mode; otherwise, it does not. The default behavior is to use Safari to display web content. You can determine whether a webpage is displayed in full-screen mode using the window.navigator.standalone read-only Boolean JavaScript property.

- content设置 yes 进入全屏模式
- 默认会启动 Safari 应用，使用 Safari 应用浏览
- 通过检测 window.navigator.standalone 的 Boolean 值可以判断 web 应用是否处于全屏模式

apple-mobile-web-app-status-bar-style

为 webapp 设置状态栏样式

```
<meta name="apple-mobile-web-app-status-bar-style" content="black">
```

This meta tag has no effect unless you first specify full-screen mode as described in apple-mobile-web-app-capable.

If content is set to default, the status bar appears normal. If set to black, the status bar has a black background. If set to black-translucent, the status bar is black and translucent. If set to default or black, the web content is displayed below the status bar. If set to black-translucent, the web content is displayed on the entire screen, partially obscured by the status bar. The default value is default.

- 此 meta 设置只在全屏模式生效
- 默认值是 default
- content="black"，状态栏背景黑色，网页内容在状态栏下面
- content="black-translucent"，状态栏半透明，背景黑色，网页内容占满全屏

该设置在 iOS6 和 iOS7 表现还可以，但到了 iOS8 后会出现各种问题，而且在 iOS9 中并没有生效。

参阅: [iOS 8: web app status bar position and resizing problems](#)

format-detection

自动识别页面中有可能是电话格式的数字

```
<meta name="format-detection" content="telephone=no">
```

By default, Safari on iOS detects any string formatted like a phone number and makes it a link that calls the number. Specifying telephone=no disables this feature.

iOS中的 Safari 会默认识别与电话格式相似的数字并生成一个可以拉起电话应用并将该数字作为电话号码拨打的链接。定义 telephone=no 可以屏蔽该功能

更多 WebApp 设置参考 [Configuring Web Applications](#)

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间： 2018-06-04 02:34:32

图片规范

图片格式

常见的图片格式有 GIF、PNG8、PNG24、JPEG、WEBP，根据图片格式的特性和场景需要选取适合的图片格式

GIF

GIF图象是基于颜色列表的（存储的数据是该点的颜色对应于颜色列表的索引值），最多只支持8位（256色）。GIF文件内部分成许多存储块，用来存储多幅图象或者是决定图象表现行为的控制块，用以实现动画和交互式应用。GIF文件还通过LZW压缩算法压缩图象数据来减少图象尺寸

特性

- 优秀的压缩算法使其在一定程度上保证图像质量的同时将体积变得很小。
- 可插入多帧，从而实现动画效果。
- 可设置透明色以产生对象浮现于背景之上的效果。
- 由于采用了8位压缩，最多只能处理256种颜色，故不宜应用于真彩色图片。

更多关于GIF：

[维基百科 - PNG](#)

[GIF文档](#)

PNG

PNG是20世纪90年代中期开始开发的图像文件存储格式，其目的是企图替代GIF和TIFF文件格式，同时增加一些GIF文件格式所不具备的特性。流式网络图形格式(Portable Network Graphic Format, PNG)名称来源于非官方的“PNG's Not GIF”，是一种位图文件(bitmap file)存储格式，读成“ping”。PNG用来存储灰度图像时，灰度图像的深度可多到16位，存储彩色图像时，彩色图像的深度可多到48位，并且还可存储多到16位的 α 通道数据。PNG使用从LZ77派生的无损数据压缩算法。

特性

- 支持256色调色板技术，文件体积小。
- 无损压缩
- 最高支持48位真彩色图像以及16位灰度图像。
- 支持Alpha通道的透明/半透明特性。
- 支持图像亮度的Gamma校准信息。
- 支持存储附加文本信息，以保留图像名称、作者、版权、创作时间、注释等信息。
- 渐近显示和流式读写，适合在网络传输中快速显示预览效果后再展示全貌。
- 使用CRC防止文件出错。
- 最新的PNG标准允许在一个文件内存储多幅图像。

更多关于PNG:

[PNG官方站 - PNG General Information](#)

[PNG格式](#)

[维基百科 - PNG](#)

JPEG

JPEG是一种针对照片视频而广泛使用的一种有损压缩标准方法。这个名称代表Joint Photographic Experts Group（联合图像专家小组）。此团队创立于公元1986年，1992年发布了JPEG的标准而在1994年获得了ISO 10918-1的认定

特性

- 适用于储存24位元全采影像
- 采取的压缩方式通常为有损压缩
- 不支持透明或动画
- 压缩比越高影像耗损越大，失真越严重
- 压缩比在10左右肉眼无法辨出压缩图与原图的差别

更多关于JPEG:

[维基百科 - JPEG](#)

WEBP

WebP，是一种同时提供了有损压缩与无损压缩的图片文件格式，派生自视频编码格式 VP8，是由Google在购买On2 Technologies后发展出来。WebP最初在2010年发布，2011年11月8日，Google开始让WebP支持无损压缩和透明色的功能，而在2012年8月16日的参考实做libwebp 0.2.0中正式支持

特性

- 同时提供有损压缩和无损压缩两种图片文件格式
- 文件体积小，无损压缩后，比 PNG 文件少了 45% 的文件大小；有损压缩后，比 JPEG 文件少了 25% - 34% 文件大小
- 浏览器兼容差，目前只支持客户端 Chrome 和 Opera 浏览器以及安卓原生浏览器(Andriod 4.0+)，[WebP兼容性](#)

更多关于WebB:

[维基百科](#)

[WebP探寻之路](#)

团队约定

内容图

内容图多以商品图等照片类图片形式存在，颜色较为丰富，文件体积较大

- 优先考虑 JPEG 格式，条件允许的话优先考虑 WebP 格式
- 尽量不使用PNG格式，PNG8 色位太低，PNG24 压缩率低，文件体积大

背景图

背景图多为图标等颜色比较简单、文件体积不大、起修饰作用的图片

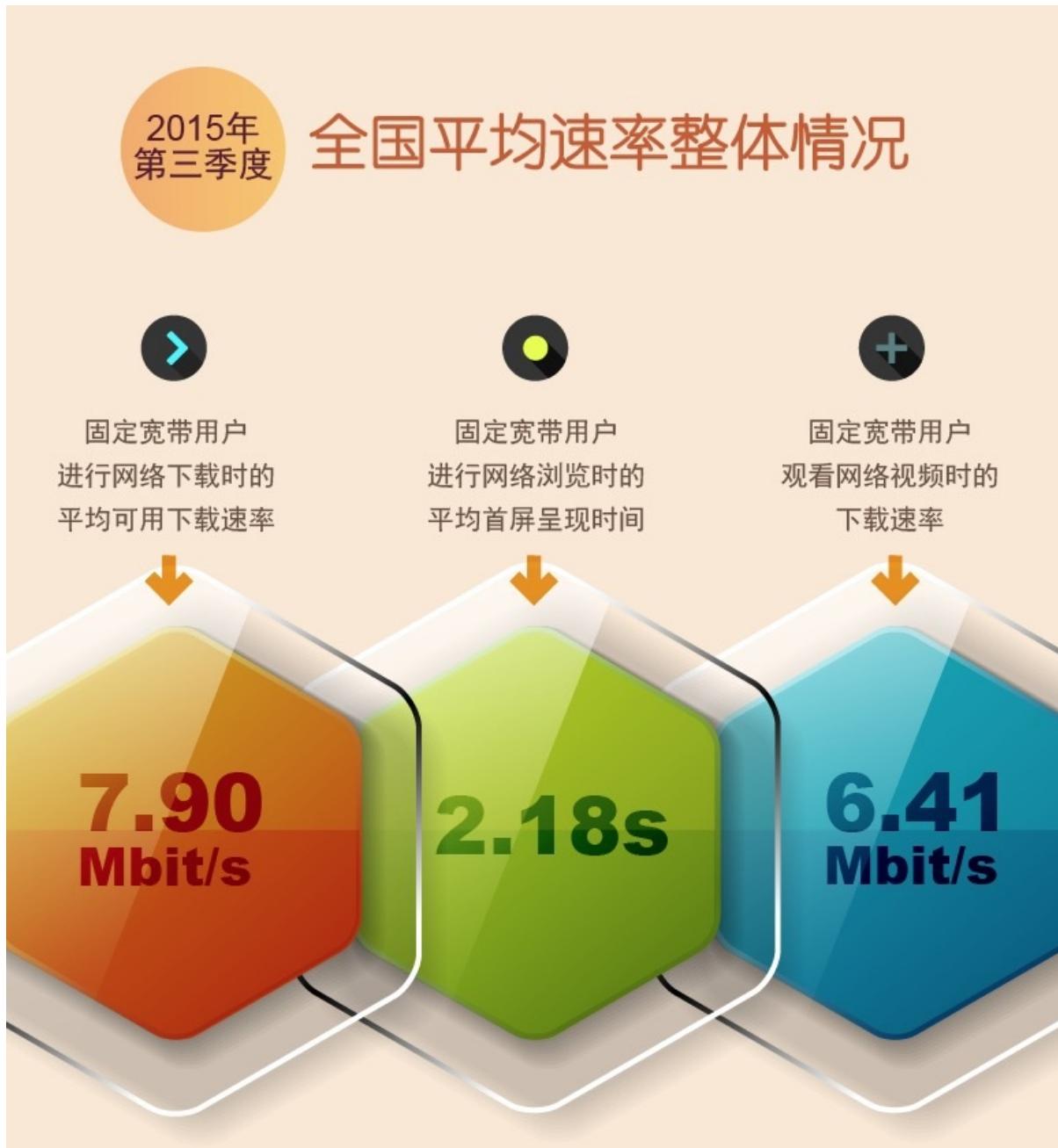
- PNG 与 GIF 格式，优先考虑使用 PNG 格式,PNG格式允许更多的颜色并提供更好的压缩率
- 图像颜色比较简单的，如纯色块线条图标，优先考虑使用 PNG8 格式，避免不使用 JPEG 格式
- 图像颜色丰富而且图片文件不太大的（40KB 以下）或有半透明效果的优先考虑 PNG24 格式
- 图像颜色丰富而且文件比较大的（40KB - 200KB）优先考虑 JPEG 格式
- 条件允许的，优先考虑 WebP 代替 PNG 和 JPEG 格式

图片大小

全国网速现状

固定网络

据文章 [《2015年Q3中国宽带速率状况报告》](#)，2015第三季全国平均速度整体情况：



中国固定宽带互联网网络平均网络下载速率达到7.90 Mbit/s，用户进行网页浏览的平均首屏呈现时间为2.18s，平均视频下载速率为6.41Mbit/s

移动网络

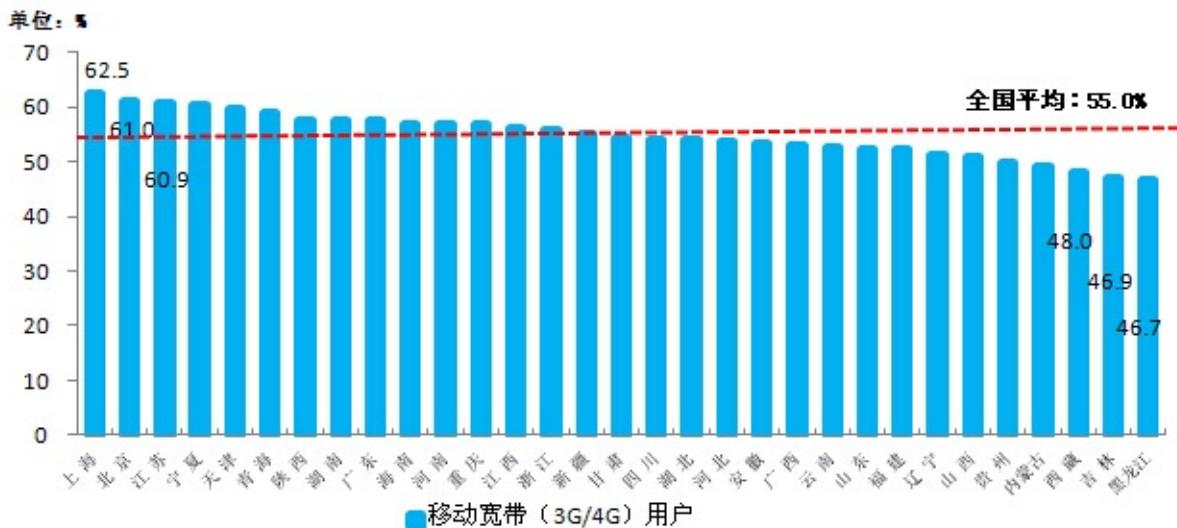
3G网络传输速率理论峰值在3.5Mbps，4G网络传输速率理论上可达到20Mbps，最高可以达到100Mbps。根据 $128KB/s = 128 \times 8 (Kb/s) = 1024Kb/s = 1Mb/s$ 的转换来算，3G网络的理论传输速率可达到450KB/s，4G网络的理论传输速率可达到 2.5MB/s ~ 12.5MB/s

受用户计算机性能、网络设备质量、资源使用情况、网络高峰期、网站服务能力、线路衰耗、信号衰减等多因素影响，3G和4G的实际平均传输速率约为：

- 3G：最高值100KB/s，平均值40~50KB/s
- 4G：最高值2.75MB/s，平均500~1000KB/s

3G/4G用户占比

2015年8月移动宽带（3G/4G）用户占比各省分布情况：





HTML 中图片引入不需添加 width、height 属性， alt 属性应该写上

推荐：

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

不推荐：

```
<img src="" width="" height="" >
```

CSS 中图片引入不需要引号

```
.jdc {  
    background-image: url(icon.png);  
}
```

CSS Sprites VS Data URIs

CSS Sprites特点

- 减少请求数
- 加速图片的显示
- 维护更新成本大
- 更多的内存消耗，特别是大体积或有过多空白的 Sprites 图
- 图片渗漏，相邻的不需展示的图片有可能出现在展示元素中，特别是在高清设备移动设备上

Data URIs (base64编码)

- 减少请求数
- 转换文件体积大，大约比原始的二进制大33%
- IE6 / IE7 不支持
- 图片显示相对较慢，需要更多的CPU消耗

更多关于 CSS Sprites 和 Data URIs 可以阅读：

[《When to Base64 Encode Images \(and When Not To\)》](#)

[《Data URI 最佳实践》](#)

[《Data URI&MHTML: 用还是不用? 》](#)

[CSS Sprites vs. Data URIs: Which is Faster on Mobile?](#)

团队约定

CSS Sprites 使用建议

- 适合使用频率高更新频率低的小图标
- 尽量不留太多的空白
- 体积较大的图片不合并
- 确保要合并的小图坐标数值和合并后的Sprites图尺寸均为偶数

Data URIs (base64编码) 使用建议

- 合适更新频率高的小图片，如某些具备自定义功能的标题icon等
- 转换成 Base64 编码的图片应小于 2KB
- 移动端不使用 Base64 编码
- 要兼容 IE6/IE7 的不使用

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间：2018-06-04
02:34:32

title: 图片格式

图片格式

常见的图片格式有 GIF、PNG8、PNG24、JPEG、WEBP，根据图片格式的特性和场景需要选取适合的图片格式。

GIF

GIF图象是基于颜色列表的（存储的数据是该点的颜色对应于颜色列表的索引值），最多只支持8位（256色）。GIF文件内部分成许多存储块，用来存储多幅图象或者是决定图象表现行为的控制块，用以实现动画和交互式应用。GIF文件还通过LZW压缩算法压缩图象数据来减少图象尺寸

特性

- 优秀的压缩算法使其在一定程度上保证图像质量的同时将体积变得很小。
- 可插入多帧，从而实现动画效果。
- 可设置透明色以产生对象浮现于背景之上的效果。
- 由于采用了8位压缩，最多只能处理256种颜色，故不宜应用于真彩色图片。

更多关于GIF：

[维基百科 - GIF](#)

[GIF文档](#)

PNG

PNG是20世纪90年代中期开始开发的图像文件存储格式，其目的是企图替代GIF和TIFF文件格式，同时增加一些GIF文件格式所不具备的特性。流式网络图形格式(Portable Network Graphic Format, PNG)名称来源于非官方的“PNG's Not GIF”，是一种位图文件(bitmap file)存储格式，读成“ping”。PNG用来存储灰度图像时，灰度图像的深度可多到16位，存储彩色图像时，彩色图像的深度可多到48位，并且还可存储多到16位的α通道数据。PNG使用从LZ77派生的无损数据压缩算法。

特性

- 支持256色调色板技术，文件体积小。
- 无损压缩
- 最高支持48位真彩色图像以及16位灰度图像。
- 支持Alpha通道的透明/半透明特性。
- 支持图像亮度的Gamma校准信息。
- 支持存储附加文本信息，以保留图像名称、作者、版权、创作时间、注释等信息。
- 渐近显示和流式读写，适合在网络传输中快速显示预览效果后再展示全貌。
- 使用CRC防止文件出错。

- 最新的PNG标准允许在一个文件内存储多幅图像。

更多关于PNG:

[PNG官方站 - PNG General Information](#)

[PNG格式](#)

[维基百科 - PNG](#)

JPEG

JPEG是一种针对照片视频而广泛使用的一种有损压缩标准方法。这个名称代表Joint Photographic Experts Group（联合图像专家小组）。此团队创立于公元1986年，1992年发布了JPEG的标准而在1994年获得了ISO 10918-1的认定

特性

- 适用于储存24位元全采影像
- 采取的压缩方式通常为有损压缩
- 不支持透明或动画
- 压缩比越高影像耗损越大，失真越严重
- 压缩比在10左右肉眼无法辨出压缩图与原图的差别

更多关于JPEG:

[维基百科 - JPEG](#)

WEBP

WebP，是一种同时提供了有损压缩与无损压缩的图片文件格式，派生自视频编码格式 VP8，是由Google在购买On2 Technologies后发展出来。WebP最初在2010年发布，2011年11月8日，Google开始让WebP支持无损压缩和透明色的功能，而在2012年8月16日的参考实做libwebp 0.2.0中正式支持

特性

- 同时提供有损压缩和无损压缩两种图片文件格式
- 文件体积小，无损压缩后，比PNG文件少了45%的文件大小；有损压缩后，比JPEG文件少了25% - 34%文件大小
- 浏览器兼容差，目前只支持客户端Chrome和Opera浏览器以及安卓原生浏览器(Andriod 4.0+)，[WebP兼容性](#)

更多关于WebP:

[维基百科 - WEBP](#)

[WEBP探寻之路](#)

团队约定

内容图

内容图多以商品图等照片类图片形式存在，颜色较为丰富，文件体积较大

- 优先考虑 JPEG 格式，条件允许的话优先考虑 WebP 格式
- 尽量不使用PNG格式，PNG8 色位太低，PNG24 压缩率低，文件体积大

背景图

背景图多为图标等颜色比较简单、文件体积不大、起修饰作用的图片

- PNG 与 GIF 格式，优先考虑使用 PNG 格式,PNG格式允许更多的颜色并提供更好的压缩率
- 图像颜色比较简单的，如纯色块线条图标，优先考虑使用 PNG8 格式，避免不使用 JPEG 格式
- 图像颜色丰富而且图片文件不太大的（40KB 以下）或有半透明效果的优先考虑 PNG24 格式
- 图像颜色丰富而且文件比较大的（40KB - 200KB）优先考虑 JPEG 格式
- 条件允许的，优先考虑 WebP 代替 PNG 和 JPEG 格式

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间：2018-06-04
02:34:32

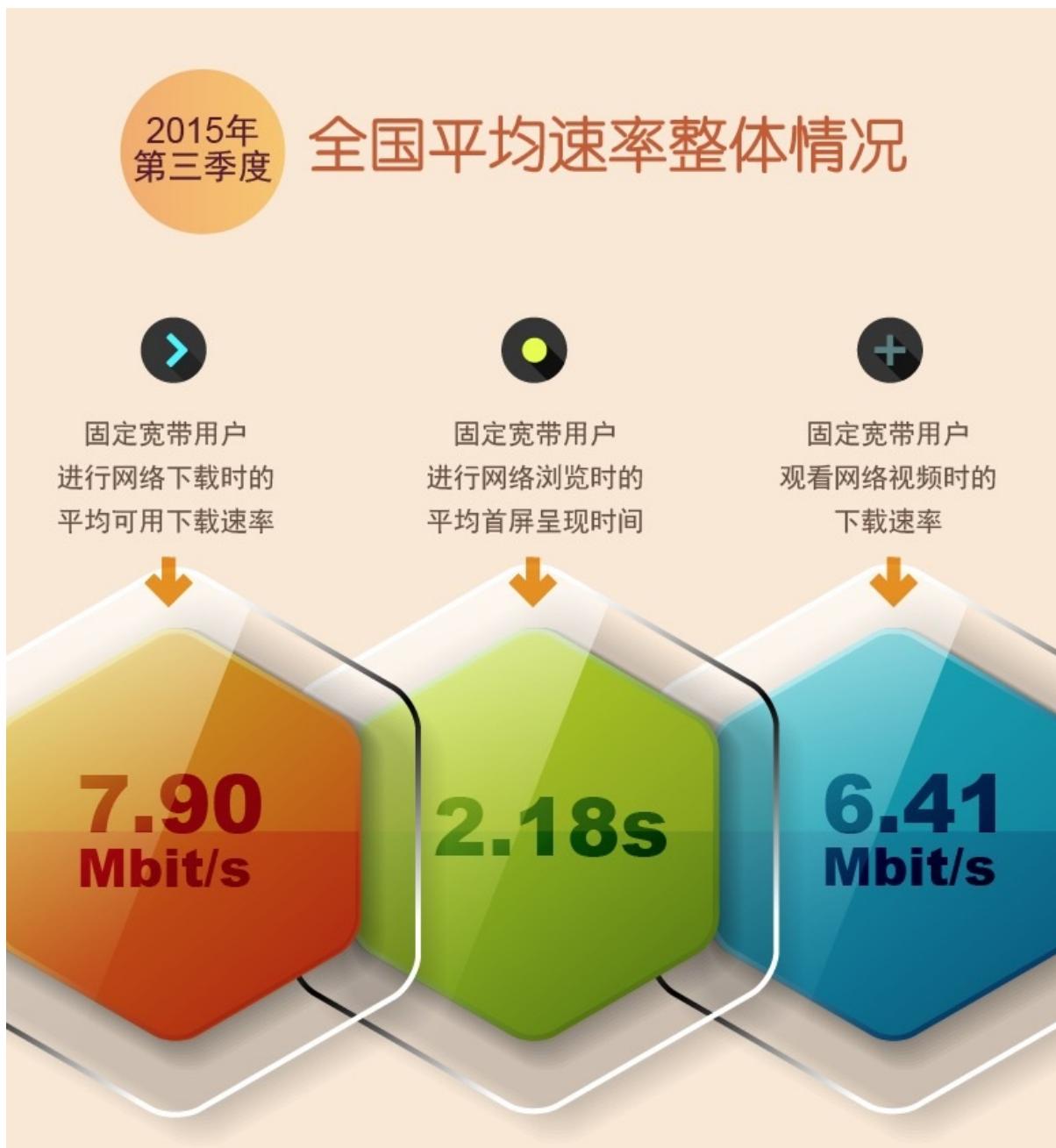
title: 图片大小

图片大小

全国网速现状

固定网络

据文章 [《2015年Q3中国宽带速率状况报告》](#)，2015第三季全国平均速度整体情况：



中国固定宽带互联网网络平均网络下载速率达到7.90 Mbit/s，用户进行网页浏览的平均首屏呈现时间为2.18s，平均视频下载速率为6.41Mbit/s

移动网络

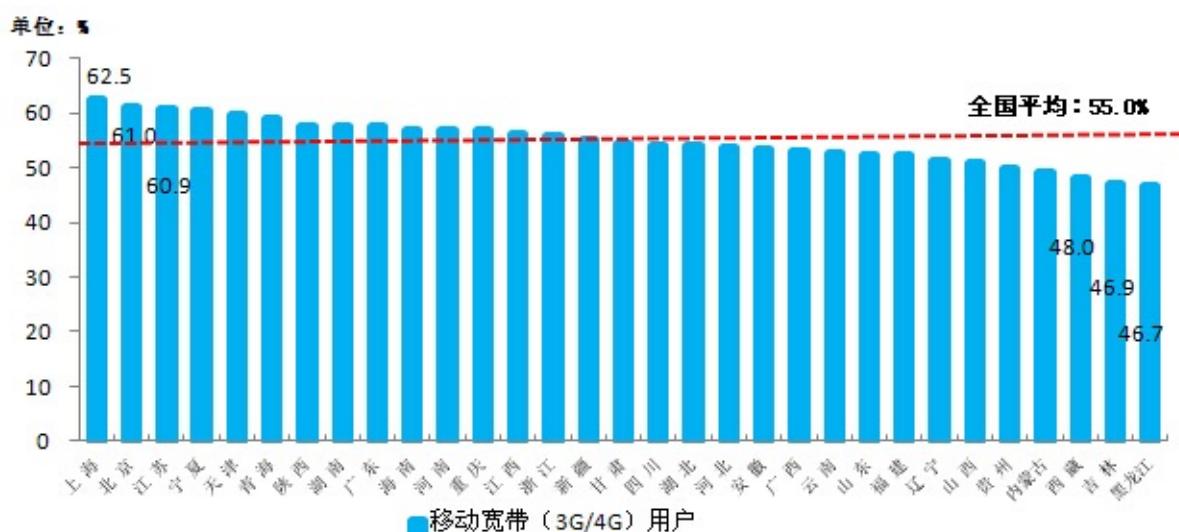
3G网络传输速率理论峰值在3.5Mbps，4G网络传输速率理论上可达到20Mbps，最高可以达到100Mbps。根据 $128\text{KB/s} = 128 \times 8(\text{Kb/s}) = 1024\text{Kb/s} = 1\text{Mb/s}$ 的转换来算，3G网络的理论传输速率可达到450KB/s，4G网络的理论传输速率可达到 2.5MB/s ~ 12.5MB/s

受用户计算机性能、网络设备质量、资源使用情况、网络高峰期、网站服务能力、线路衰耗、信号衰减等多因素影响，3G和4G的实际平均传输速率约为：

- 3G: 最高值100KB/s, 平均值40~50KB/s
 - 4G: 最高值2.75MB/s, 平均500~1000KB/s

3G/4G用户占比

2015年8月移动宽带（3G/4G）用户占比各省分布情况：



据文章《工信部：2015年7月底中国4G用户累计超过2.5亿》介绍：

截至2015年7月底，中国4G用户累计超过2.5亿（全球LTE用户超过7.9亿，全球TD-LTE用户超过2.78亿），已建设4G基站超过153万个，其中完成TD-LTE基站建设超过100万个，多载波聚合等TD-LTE演进技术逐步商用，4G智能手机已经占到国内智能手机市场的82.7%。

团队约定

中国普通家庭的宽带基本能达到8Mbps，实际速率大约为500--900KB/s，全国3G/4G用户占有比超过了50%，为了保证图片能更好地加载展示给用户看，团队约定：

PC平台单张的图片的大小不应大于 200KB。

移动平台单张的图片的大小不应大于 100KB。

(图片的大小约定标准随全国网速的改变而改变)

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间： 2018-06-04
02:34:32

title: 图片质量

- 上线的图片都应该经过压缩处理，压缩后的图片不应该出现肉眼可感知的失真区域
- 60质量的JPEG格式图片与质量大于60的相比，肉眼已看不出明显的区别，因此保存 JPEG 图的时候，质量一般控制在60，若保真度要求高的图片可适量提高到 80，图片大小控制在 200KB 以内

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间：2018-06-04
02:34:32

title: 图片引入

图片引入

测试内容图应该写上表明图片尺寸的占位图，可以用线上占位图生成服务，如：

```
http://placeholder.qiniudn.com/300x200
```



HTML 中图片引入不需添加 width、height 属性， alt 属性应该写上

推荐：

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

不推荐：

```
<img src="" width="" height="" >
```

CSS 中图片引入不需要引号

```
.jdc {  
    background-image: url(icon.png);  
}
```

CSS Sprites VS Data URIs

CSS Sprites特点

- 减少请求数
- 加速图片的显示
- 维护更新成本大
- 更多的内存消耗，特别是大体积或有过多空白的 Sprites 图
- 图片渗漏，相邻的不需展示的图片有可能出现在展示元素中，特别是在高清设备移动设备上

Data URIs (base64编码)

- 减少请求数
- 转换文件体积大，大约比原始的二进制大33%
- IE6 / IE7 不支持
- 图片显示相对较慢，需要更多的CPU消耗

更多关于 CSS Sprites 和 Data URIs 可以阅读：

[《When to Base64 Encode Images \(and When Not To\)》](#)

[《Data URI 最佳实践》](#)

[《Data URI&MHTML: 用还是不用? 》](#)

[CSS Sprites vs. Data URIs: Which is Faster on Mobile?](#)

团队约定

CSS Sprites 使用建议

- 适合使用频率高更新频率低的小图标
- 尽量不留太多的空白
- 体积较大的图片不合并
- 确保要合并的小图坐标数值和合并后的 Sprites 图尺寸均为偶数

Data URIs (base64编码) 使用建议

- 适合更新频率高的小图片，如某些具备自定义功能的标题icon等
- 转换成 Base64 编码的图片应小于 2KB
- 移动端不使用 Base64 编码
- 要兼容 IE6/IE7 的不使用

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间：2018-06-04
02:34:32

CSS规范

代码规范

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.
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

关于 @charset

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，在可控范围内，坚决不使用 BOM。（更多关于 BOM 可参考 [BOM的介绍](#) 和 [「带 BOM 的 UTF-8」和「无 BOM 的 UTF-8」有什么区别？](#)）

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](#)

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 {
```

```
    ...
}

.nav{
    ...
}
```

不推荐:

```
.jdc, jdc_logo, .jdc_hd {
    ...
}

.nav{
    ...
}
```

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](#)

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缀写法：[#Vendor-specific extensions](#)

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
- 注释不能被嵌套

```
/*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](#)

SASS规范

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](#)。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](#)

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 会假定文档的编码为 UTF-8

团队约定

严格遵守上面“代码规范”中的“编码规范”

更多关于 SASS 编码：[SASS Encodings](#)

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，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 {
```

```
// CSS
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

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; font-family: Helvetica, STHeiti STXihei, Microsoft JhengHei, Microsoft YaHei, Arial; 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](#)

媒体查询类型浏览器支持情况：[CSS3 Media Queries overview](#)

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) {  
  
}
```

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间： 2018-06-04
02:34:32

title: 代码规范

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.
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，在可控范围内，坚决不使用 BOM。（更多关于 BOM 可参考 [BOM的介绍](#) 和 [「带 BOM 的 UTF-8」和「无 BOM 的 UTF-8」有什么区别？](#)）

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](#)

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](#)

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缀写法：[#Vendor-specific extensions](#)

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间：2018-06-04
02:34:32

title: 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](#)

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间：2018-06-04 02:34:32

title: SASS规范

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](#)。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](#)

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”

更多关于 SASS 编码：[SASS Encodings](#)

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，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

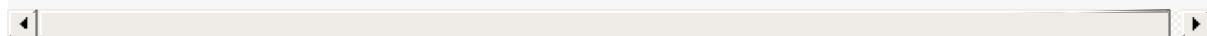
```
// 正确的运算格式
.jdc {
    width: 100px - 50px;
    width: 100px + 50px;
    width: 100px * 2;
    width: 100px / 2;
}
```

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间：2018-06-04
02:34:32

title: 重置样式

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; font-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, hgroup, 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; }
```



Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间： 2018-06-04
02:34:32

title: 媒体查询

设备尺寸参考：[Mobile devices](#)

媒体查询类型浏览器支持情况：[CSS3 Media Queries overview](#)

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) {  
  
}
```

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook该文件修订时间：2018-06-04
02:34:32

title: 移动端常用私有属性

目前两大主流移动平台为 iOS 和 Android，有不少带 -webkit- 前缀的 CSS 私有属性以及一些 iOS only 属性，当中好些属性在日常需求中经常应用到。

WebKit CSS 属性中的一部分已经被包含在 CSS 规范草案中，并且可能成为最后的推荐标准，但目前仍然是试验性的属性，还有一些属性是不规范的属性，它们没有出现在跟踪规范中。

-webkit-scrollbar

`-webkit-scrollbar` 是-webkit-私有的伪元素，用于对拥有overflow属性的区域 自定义滚动条的样式。

譬如，为了隐藏滚动条，你可以这么做：

```
.scroll::-webkit-scrollbar {
    width: 0;
    height: 0;
}
```

除了对整个滚动条的控制外，Webkit还提供了控制对滚动条各组成部分的表现渲染的伪元素，甚至具体到滚动条的各种状态行为的伪类。

滚动条各块组成表现渲染的伪元素

一般而言，滚动条的主要组成部分包括：

- 滚动按钮 — 滚动按钮的夹角则被称为滚动角(corner)。
- 轨道 — 轨道(track)可以进一步分为轨枕(track pieces) 和滑块(thumb)。

Webkit则根据滚动条各组成部分，提供了不同的伪元素来自定义样式。

```
::-webkit-scrollbar          { /* 1 */ }
::-webkit-scrollbar-button   { /* 2 */ }
::-webkit-scrollbar-track    { /* 3 */ }
::-webkit-scrollbar-track-piece { /* 4 */ }
::-webkit-scrollbar-thumb    { /* 5 */ }
::-webkit-scrollbar-corner   { /* 6 */ }
::-webkit-resizer            { /* 7 */ }
```

下图则是各伪元素对应的滚动条各部分：



`::-webkit-scrollbar` : 滚动条整体部分。可设置width、height、background、border等。

`::-webkit-scrollbar-button` : 滚动条两端的按钮。可以用`display:none`让其不显示，也可以添加背景图片，颜色改变显示效果。

`::-webkit-scrollbar-track` : 轨道。可以用`display:none`让其不显示，也可以添加背景图片，颜色改变显示效果。

`::-webkit-scrollbar-track-piece` : 轨枕，也就是除去滚动滑块的部分。

`::-webkit-scrollbar-thumb` : 滚动滑块，也就是滚动条里面可以拖动的那部分。

`::-webkit-scrollbar-corner` : 滚动按钮的夹角则被称为滚动角。

`::-webkit-resizer` : 用于定义右下角拖动块的样式。

需要注意的是：若是水平滚动条，则`width`属性不起作用，`height`属性用来控制滚动条相应部分竖直方向高度；若是竖直滚动条，则`height`属性不起作用，`width`属性用来控制相应部分的宽度。

滚动条各块组成的伪元素

下面的伪类可以应用到上面的伪元素中。`:horizontal` : 选择水平方向的滚动条。

`:vertical` : 选择垂直方向的滚动条。

`:decrement` : 适用于滚动按钮和轨枕。选择能够使得视窗位置递减状态(例如，垂直滚动条向上滚动，水平滚动条向左滚动。)的滚动按钮或轨枕。

`:increment` : 适用于滚动按钮和轨枕。选择能够使得视窗位置递增状态(例如，垂直滚动条向下滚动，水平滚动条向右滚动。)的滚动按钮或轨枕。

`:start` : 适用于滚动按钮和轨枕。选择位于滚动滑块前边的滚动按钮和轨枕。

`:end` : 适用于滚动按钮和轨枕。选择位于滚动滑块后边的滚动按钮和轨枕。

`:double-button` : 适用于滚动按钮和轨枕。选中紧挨着一对按钮的轨枕以及位于滚动条某一端的一对按钮中的其中一个滚动按钮。

`:single-button` : 适用于滚动按钮和轨枕。选中紧挨着仅一个按钮的轨枕以及位于滚动条某一端的仅它本身一个的滚动按钮。

`:no-button` : 适用于轨枕。选中轨道结束位置没有按钮的轨枕。

`:corner-present` : 适用于选中滚动角不存在的滚动条。

`:window-inactive` : 适用于所有滚动条，选中焦点不在该视窗的滚动区域。

另外，`:enabled`、`:disabled`、`:hover`、和`:active`等伪类同样在滚动条中适用。为了更好地理解，以下是几个伪元素组合伪类的应用例子：

```
::-webkit-scrollbar-track-piece:start {
    /*滚动条上半边或左半边*/
}
::-webkit-scrollbar-thumb>window-inactive {
```

```
/*当焦点不在当前区域滑块的状态*/
::-webkit-scrollbar-button:horizontal:decrement:hover {
    /*当鼠标在水平滚动条下面的按钮上的状态*/
}
```

或者，读者可以去阅读[官方例子](#)

参考资料：

- [Webkit-Styling Scrollbars](#)
- [自定义浏览器滚动条的样式，打造属于你的滚动条风格](#)

-webkit-touch-callout

`-webkit-touch-callout` 是一个不规范的属性 ([unsupported WebKit property](#))，它没有出现在 CSS 规范草案中。

当你触摸并按住触摸目标时候，禁止或显示系统默认菜单。在iOS上，当你触摸并按住触摸的目标，比如一个链接，Safari浏览器将显示链接有关的系统默认菜单，这个属性可以让你禁用系统默认菜单。

属性值

- `none`：系统默认菜单被禁用
- `inherit`：系统默认菜单不被禁用

兼容性

- iOS 2.0及更高版本的 Safari 浏览器可用
- Android 尚不明确

-webkit-tap-highlight-color

`-webkit-tap-highlight-color` 是一个不规范的属性 ([unsupported WebKit property](#))，它没有出现在 CSS 规范草案中。

在 iOS Safari 上，当用户点击链接或具有 JavaScript 可点击脚本的元素，系统会为这些被点击元素加上一个默认的透明色值，该属性可以覆盖该透明值。

属性值

`<color>`：颜色值

兼容性

- iOS 1.1.1及更高版本的Safari浏览器可用

- 大部分安卓手机

-webkit-overflow-scrolling

定义在具 `overflow:scroll` 属性的元素内是否采用原生样式滚动行为

属性值

- `auto` : 默认值，单手滚动，滚动没有惯性效果
- `touch` : 原生样式滚动，应用此属性值会产生层叠上下文（会影响定位参照物的属性，类似 `opacity`、`masks`、`transforms` 属性，影响到 `position` 的效果，甚至影响到 `position:fixed` 的定位参照物，）

兼容性

- iOS 5.0 及更高版本
- 大部分安卓机

-webkit-line-clamp

`-webkit-line-clamp` 是一个不规范的属性 ([unsupported WebKit property](#))，它没有出现在 CSS 规范草案中。

限制在一个块元素显示的文本的行数。为了实现该效果，它需要组合其他外来的WebKit属性。

常见结合属性：

- `display: -webkit-box` : 必须结合的属性，将对象作为弹性伸缩盒子模型显示。
- `-webkit-box-orient` : 必须结合的属性，设置或检索伸缩盒对象的子元素的排列方式。
- `text-overflow` : 可以用来多行文本的情况下，用省略号“...”隐藏超出范围的文本。

属性值

`<number>`: 块元素显示的文本的行数

兼容性

- iOS
- Andriod

-webkit-appearance

`-webkit-appearance` 是一个不规范的属性 ([unsupported WebKit property](#))，它没有出现在 CSS 规范草案中。

改变按钮和其他控件的外观，使其类似于原生控件。

属性值

- `none` : 去除系统默认 appearance 的样式，常用于 iOS 下移除原生样式
- `button` : 渲染成 button 的风格
- `checkbox` : 渲染成 input checkbox 样式的复选框按钮
- `radio` : 渲染成 radio 的风格
- ...

更多属性值参考 [mozilla: -webkit-appearance 属性](#)

兼容性

- iOS 2.0及更高版本的Safari浏览器可用
- Android 尚不明确

-webkit-font-smoothing

字体平滑，该属性定义文本的平滑属性，但要注意以下说明：

非标准属性，不建议用于网页上，这个属性不能保证所有用户都能看到相同效果，这会使网站的字体渲染造成不一致，而此属性的渲染行为日后也有可能会改变

属性值

- `none` : 去掉字体平滑效果，使字体带锯齿
- `antialiased` : 使字体在像素级别更平滑更轻细
- `subpixel-antialiased` : 在多数非 Retina 显示设备中字体将会更锐利。

注意：以上属性在 Retina 设备上会有明显的效果，在非 Retina 设备上看不出差异

兼容性

- 部分高清设备，如 Retina Mac

-webkit-backface-visibility

`backface-visibility` 在 [W3文档](#) 有定义描述

定义转换元素的背面是否显示

属性值

- `visible` : 显示（默认值）

- `hidden` : 隐藏

兼容性

- iOS 2.0 及更高版本的 Safari 浏览器可用
- 大部分 Android

-webkit-mask

定义多样的蒙板效果属性（缩写属性，类似 `margin`）

使用语法

```
<mask-image> [<mask-repeat> || <mask-attachment> || <mask-position> || <mask-origin> || <mask-clip> || <mask-composite>]*  
where  
<mask-position> = [ <percentage> | <length> | left | center | right ] [ <percentage> | <length> | top | center | bottom ]?
```

默认值：

```
-webkit-mask: none repeat scroll 0% 0% padding border add;
```

属性值

- `<mask-image>`: 为元素设置蒙板图片，蒙板图片会根据图片的透明区域对元素可视部分进行裁剪
 - `<uri>`: 图片链接作为蒙板图片
 - `<gradient>`: 渐变函数 `-webkit-gradient` 作为蒙板图片
 - `none` : 去掉蒙板图片
- `<mask-repeat>`: 定义蒙板图片是否平铺或平铺的方式
 - `repeat` : 默认值，水平和垂直方向平铺
 - `repeat-x` : 水平方向平铺
 - `repeat-y` : 垂直方向平铺
 - `no-repeat` : 不平铺
- `<mask-attachment>`: 如果 `-webkit-mask-image` 属性有设置，`attachment` 决定该图片是否相对视窗固定或随着其容器滚动
 - `scroll` : 默认值，随容器滚动
 - `fixed` : 相对视窗固定
- `<mask-position>`: 定义蒙板图片的初始位置，书写格式类似 `background-position` ---- `<mask-position>[, <mask-position>]*`
 - `<percentage>`

- <length>
 - left
 - right
 - center
- <mask-origin>：定义蒙板图片定位相对起点，与 `webkit-mask-position` 属性相关。当 `-webkit-mask-attachment:fixed` 的时候，该属性不生效。
 - padding : 默认值，蒙板定位相对边距
 - border : 蒙板定位相对边框
 - content : 蒙板定位相对元素盒子内容
 - <mask-clip>：如果 `-webkit-mask-image` 属性有设置，`-webkit-mask-clip` 将定义蒙板图片的裁剪区域
 - border : 默认值，蒙板图片延伸到容器的边框
 - padding : 蒙板图片延伸到容器的边距
 - content : 蒙板图片裁剪到元素盒子内容范围
 - text : 蒙板图片裁剪到元素文本范围
 - <mask-composite>：定义蒙板图片重合的裁剪显示方式
 - add : 默认值，图片重合不裁剪
 - subtract : 去掉层级低的图形以及图片重合部分图形，只留层级高非重合部分图形
 - intersect : 只留重合部分图形
 - exclude : 只去掉重合部分图形

有关属性更详细描述请参考：

[w3 - css-masking](#)

[MDN - -webkit-mask](#)

[携程 UED - -webkit-mask](#)

兼容性

- Safari 4.0 及更高版本
- iOS 4.0 及更高版本
- Android 2.1 及更高版本

-webkit-user-select

定义用户是否能选中元素内容

属性值

- auto : 可选中元素内容
- none : 不能选中任何内容

- `text` : 可选中元素内的文本

兼容性

- iOS 3.0 及更高版本的 Safari
- 大部分安卓手机

-webkit-user-modify

定义用户是否可编辑元素内容

属性值

- `read-only` : 只读
- `read-write` : 可读可写, 粘贴内容会保留富文本格式 (Android 机不保留富文本格式)
- `read-write-plaintext-only` : 可读可写, 粘贴内容所有富文本格式都会丢失

注意: 使用这个属性的时候, 请不要出现 `-webkit-user-select: none`, 文本无法选中的情况下, 在 Safari 该属性不生效, 不过在 Chrome 依然生效

兼容性

- iOS 5.0 及更高版本
- Safari 3.0 及更高版本
- 大部分安卓手机

-webkit-text-stroke

定义文本描边, 可以设计描边的宽和颜色, 一般与文本填充属性 `-webkit-text-fill-color` 共用。

属性值

- `<length>`: 长度单位
- `<color>`: 颜色值

兼容性

- iOS 2.0 及更高版本
- Safari 3.0 及更高版本
- 安卓尚不明确

-webkit-text-fill-color

定义文本填充, 一般与文本描边属性 `-webkit-text-stroke` 共用。

属性值

- <color>: 颜色值
- `currentcolor` : 元素 `color` 属性值
- `-webkit-activelink` : 链接被点击时系统的默认颜色

更多属性值参考: [Safari CSS Reference -webkit-text-fill-color](#)

兼容性

- iOS 2.0 及更高版本
- Safari 3.0 及更高版本
- 安卓尚不明确

-webkit-text-size-adjust

定义 iOS Safari 网页文本大小调整属性

属性值

- <percentage>: 百分比值, 字体显示调整值
- `auto` : 字体自动调整
- `none` : 字体不能自动调整

兼容性

- iOS 1.0 及更高版本
- Safari on iOS only
- 安卓尚不明确

-webkit-marquee

定义滚动文本内容属性 (缩写属性, 类似 `margin`) 。

使用语法

```
-webkit-marquee: direction increment repetition style speed
```

属性值

- <direction>: 滚动方向
 - `ahead` : 从下到上滚动
 - `auto` : 默认滚动方向
 - `backwards` : 从右到左滚动

- down : 从上到下滚动
- forwards : 从左到右滚动
- left : 从右到左滚动
- reverse : 从上到下滚动
- right : 从左到右滚动
- up : 从下到上滚动
- <increment>: 每次移动的距离
 - [<percentage> | <length>]
 - large : 距离常量
 - medium : 距离常量
 - small : 距离常量
- <repetition>: 文字滚动的重复次数
 - 非负整数
 - infinite : 无限次
- <style>: 文字滚动的方式
 - alternate : 重复滚动
 - none : 停止滚动
 - scroll : 在定义方向上滚动
 - slide : 定义方向上滚动, 内容显示完毕或者内容滚动到滚动区域另一端边框时候都会停止下来
- <speed>: 滚动或滑动的速度
 - 非负整数 (毫秒单位) 或带时间单位的非负整数
 - fast
 - normal
 - slow

兼容性

- iOS 1.0 及更高版本
- Safari 3.0 及更高版本
- 大部分安卓手机

-webkit-filter

滤镜属性可以让元素本身内容 (文本、背景等) 及其子元素加上滤镜效果

属性值

- blur(<length>) : 模糊, 原始效果值为 0px , 不接受负值
- brightness([<number> | <percentage>]) : 亮度, 原始效果值为 1 或 100% , 不接受负值

- `contrast([<number> | <percentage>])` : 对比度, 原始效果值为 `1` 或 `100%`, 不接受负值
- `drop-shadow(<length>{2, 4} <color>?)` : 投影, 原始效果值为所有长度值为 `0`, 长度值至少2个, 最多4个,
- `grayscale([<number> | <percentage>])` : 灰度, 原始效果值为 `0`, 最大值为 `1` 或 `100%`, 不接受负值
- `hue-rotate(<angle>)` : 相位, 原始效果值为 `0deg`
- `invert([<number> | <percentage>])` : 反相, 原始效果值为 `0`, 最大值为 `1` 或 `100%`, 不接受负值
- `opacity([<number> | <percentage>])` : 透明度, 原始效果值为 `1`, 最大值为 `1` 或 `100%`, 不接受负值
- `saturate([<number> | <percentage>])` : 饱和度, 原始效果值为 `1`, 不接受负值
- `sepia([<number> | <percentage>])` : 乌贼墨, 原始效果值为 `0`, 最大值为 `1` 或 `100%`, 不接受负值

关于 `-webkit-filter` 与 `-webkit-backdrop-filter` 的属性对比可以参考:

[What's New in Safari 9.0 - backdrop-filter](#)

兼容性

- iOS 8.0 及更高版本
- Safari 8.0 及更高版本
- Android 4.4 及更高版本

-webkit-backdrop-filter

背景滤镜属性可以让元素的背景或元素层级以下的元素加上滤镜效果

属性值

- `blur(<length>)` : 模糊, 原始效果值为 `0px`, 不接受负值
- `brightness([<number> | <percentage>])` : 亮度, 原始效果值为 `1` 或 `100%`, 不接受负值
- `contrast([<number> | <percentage>])` : 对比度, 原始效果值为 `1` 或 `100%`, 不接受负值
- `drop-shadow(<length>{2, 3} <color>?)` : 投影, 原始效果值为所有长度值为 `0`, 长度值至少2个, 最多3个, 注意: 不支持投影扩展值和混合投影

- `grayscale([<number> | <percentage>])` : 灰度, 原始效果值为 `0`, 最大值为 `1` 或 `100%`, 不接受负值
- `hue-rotate(<angle>)` : 相位, 原始效果值为 `0deg`
- `invert([<number> | <percentage>])` : 反相, 原始效果值为 `0`, 最大值为 `1` 或 `100%`, 不接受负值
- `opacity([<number> | <percentage>])` : 透明度, 原始效果值为 `1`, 最大值为 `1` 或 `100%`, 不接受负值
- `saturate([<number> | <percentage>])` : 饱和度, 原始效果值为 `1`, 不接受负值
- `sepia([<number> | <percentage>])` : 乌贼墨, 原始效果值为 `0`, 最大值为 `1` 或 `100%`, 不接受负值

关于 `-webkit-filter` 与 `-webkit-backdrop-filter` 的属性对比可以参考:

[What's New in Safari 9.0](#)

兼容性

- iOS 9.0 及更高版本
- Safari 9.0 及更高版本
- 安卓尚未明确

position:-webkit-sticky

可以使得元素在页面没有滚动的情况下表现得像`relative`, 在滚动条滚到该元素区域的时候根据`top`值的设置使元素固定离顶部的距离, 表现像 `position:fixed`, 也就是常见的吸顶需求效果。

特性

- 依赖父级元素滚动区域
- 定位参考物始终是 `viewport`, `transform` 等可以改变 `position:fixed` 定位参考物的属性也无法改变 `position:-webkit-sticky` 的定位参考物
- `position:-webkit-sticky` 属性的元素固定区域只依赖其父元素的可滚动高度, 如果其父元素高度小于元素本身的高度, `fixed`效果失效。

兼容性

- iOS 6.1 及更高版本
- iOS only

-apple-system

苹果操作系统会从两种不同外观和大小的字体进行自动转换去调节系统新字体“San Francisco”，可以通过 CSS 规则

```
font-family: -apple-system , sans-serif;
```

让系统智能选择适配操作系统的字体，添加 `-apple-system` 可以使字体变得更圆润锐利。

关于 `-apple-system` 更详细的介绍可以参考：

[What's New in Safari 9.0](#)

兼容性

- iOS 9.0 及更高版本
- Safari 9.0 及更高版本
- iOS / OS X only

更多 WebKit CSS 属性

更多 `-webkit-` CSS 属性介绍请参考：

- [MDN文档 - Webkit Extensions](#)
- [携程 UED - webkitcss](#)

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间：2018-06-04
02:34:32

命名规范

由历史原因及个人习惯引起的 DOM 结构、命名不统一，导致不同成员在维护同一页面时，效率低下，迭代、维护成本极高。

目录命名

- 项目文件夹：your project name
- 样式文件夹：css
- 脚本文件夹：js
- 样式类图片文件夹：images

HTML/CSS文件命名

确保命名总是以字母开头而不是数字，且字母一律小写，以下划线连接

```
<!-- HTML -->
jdc.html
jdc_list.html
jdc_detail.html

<!-- SASS -->
jdc.scss
jdc_list.scss
jdc_detail.scss
```

图片命名

命名顺序

图片命名建议以下顺序命名：

图片业务(可选) + (mod_) 图片功能类别 (必选) + 图片模块名称 (可选) + 图片精度 (可选)

- 图片业务：
 - pp_：拍拍
 - wx_：微信
 - sq_：手Q
 - jd_：京东商城
 - ...
- 图片功能类别：
 - mod_：是否公共，可选
 - icon：模块类固化的图标
 - logo：LOGO类

- spr: 单页面各种元素合并集合
- btn: 按钮
- bg: 可平铺或者大背景
- ...

- 图片模块名称:

- goodslist: 商品列表
- goodsinfo: 商品信息
- userava tar: 用户头像
- ...

- 图片精度:

- 普清: @1x
- Retina: @2x | @3x
- ...

如下面例子：

```
公共模块:  
wx_mod_btn_goodlist@2x.png  
wx_mod_btn_goodlist.png  
mod_btn_goodlist.png
```

```
非公共模块:  
wx_btn_goodlist@2x.png  
wx_btn_goodlist.png  
btn_goodlist.png
```

交叉业务协作

业务交叉协作的时候，为了避免图片命名冲突，建议图片名加上业务和模块前缀，如拍拍侧和手Q侧的业务交叉合作时，侧栏导航icon雪碧图命名：

```
推荐:  
pp_icon_mod_sidenav.png
```

```
不推荐:  
icon_mod_sidenav.png
```

处理高清图片的时候，命名应该加上图片相应的精度说明

```
推荐:  
jdc_logo@1x.png  
jdc_logo@2x.png
```

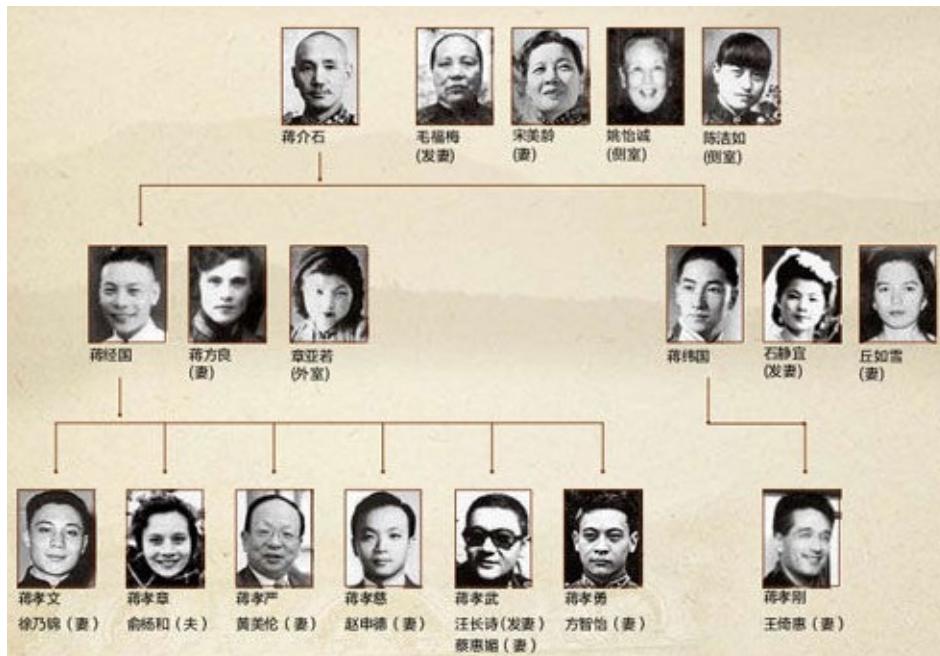
```
不推荐:  
jdc_logo.png  
jdc_logo_retina.png
```

ClassName命名

ClassName的命名应该尽量精短、明确，必须以字母开头命名，且全部字母为小写，单词之间统一使用下划线“_”连接

命名原则

基于姓氏命名法（继承 + 外来），如下图：



祖先模块不能出现下划线，除了是全站公用模块，如 `mod_` 系列的命名：

推荐：

```
<div class="modulename">
    <div class="modulename_info">
        <div class="modulename_son"></div>
        <div class="modulename_son"></div>
        ...
    </div>
</div>

<!-- 这个是全站公用模块，祖先模块允许直接出现下划线 -->
<div class="mod_info">
    <div class="mod_info_son"></div>
    <div class="mod_info_son"></div>
    ...
</div>
```

不推荐：

```
<div class="modulename_info">
    <div class="modulename_info_son"></div>
    <div class="modulename_info_son"></div>
    ...
</div>
```

在子孙模块数量可预测的情况下，严格继承祖先模块的命名前缀

```
<div class="modulename">
    <div class="modulename_cover"></div>
    <div class="modulename_info"></div>
</div>
```

当子孙模块超过4级或以上的时候，可以考虑在祖先模块内具有识别性的独立缩写作为新的子孙模块

推荐：

```
<div class="modulename">
    <div class="modulename_cover"></div>
    <div class="modulename_info">
        <div class="modulename_info_user">
            <div class="modulename_info_user_img">
                <img src="" alt="">
                <!-- 这个时候 jiui 为 modulename_info_user_img 首字母缩写-->
                <div class="jiui_tit"></div>
                <div class="jiui_txt"></div>
                ...
            </div>
        </div>
        <div class="modulename_info_list"></div>
    </div>
</div>
```

不推荐：

```
<div class="modulename">
    <div class="modulename_cover"></div>
    <div class="modulename_info">
        <div class="modulename_info_user">
            <div class="modulename_info_user_img">
                <img src="" alt="">
                <div class="modulename_info_user_img_tit"></div>
                <div class="modulename_info_user_img_txt"></div>
                ...
            </div>
        </div>
        <div class="modulename_info_list"></div>
    </div>
</div>
```

```
</div>
```

模块命名

全站公共模块：以 `mod_` 开头

```
<div class="mod_yours"></div>
```

业务公共模块：以 `业务名_mod_` 开头

```
<div class="paipai_mod_yours"></div>
```

常用命名推荐

注意：ad,baaner 等有机会和广告挂勾的字眼不建议直接用来做 `ClassName`，因为有些浏览器插件（Chrome的广告拦截插件等）会直接过滤这些类名，因此

`<div class="ad"></div>` 这种情况不应该出现

另外，**敏感不和谐字眼**也不应该出现，如

```
<div class="fuck"></div>
<div class="jer"></div>
<div class="sm"></div>
<div class="isis"></div>
<div class="KMT"></div>
...
...
```

ClassName	含义
about	关于
account	账户
arrow	箭头图标
article	文章
aside	边栏
audio	音频
avatar	头像
bg,background	背景
bar	栏（工具类）
branding	品牌化
crumb,breadcrumbs	面包屑
btn,button	按钮

caption	标题, 说明
category	分类
chart	图表
clearfix	清除浮动
close	关闭
col,column	列
comment	评论
community	社区
container	容器
content	内容
copyright	版权
current	当前态, 选中态
default	默认
description	描述
details	细节
disabled	不可用
entry	文章, 博文
error	错误
even	偶数, 常用于多行列表或表格中
fail	失败 (提示)
feature	专题
fewer	收起
field	用于表单的输入区域
figure	图
filter	筛选
first	第一个, 常用于列表中
footer	页脚
forum	论坛
gallery	画廊
group	模块, 清除浮动
header	页头
help	帮助
hide	隐藏

highlight	高亮
home	主页
icon	图标
info,information	信息
last	最后一个，常用于列表中
links	链接
login	登录
logout	退出
logo	标志
main	主体
menu	菜单
meta	作者、更新时间等信息栏，一般位于标题之下
module	模块
more	更多 (展开)
msg, message	消息
nav,navigation	导航
next	下一页
nub	小块
odd	奇数，常用于多行列表或表格中
off	鼠标离开
on	鼠标移过
output	输出
pagination	分页
pop,popup	弹窗
preview	预览
previous	上一页
primary	主要
progress	进度条
promotion	促销
rcomm,recommendations	推荐
reg,register	注册
save	保存
search	搜索

secondary	次要
section	区块
selected	已选
share	分享
show	显示
sidebar	边栏, 侧栏
slide	幻灯片, 图片切换
sort	排序
sub	次级的, 子级的
submit	提交
subscribe	订阅
subtitle	副标题
success	成功 (提示)
summary	摘要
tab	标签页
table	表格
txt, text	文本
thumbnail	缩略图
time	时间
tips	提示
title	标题
video	视频
wrap	容器, 包, 一般用于最外层
wrapper	容器, 包, 一般用于最外层

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间: 2018-06-04
02:34:32

title: 命名规范

由历史原因及个人习惯引起的 DOM 结构、命名不统一，导致不同成员在维护同一页面时，效率低下，迭代、维护成本极高。

目录命名

- 项目文件夹：projectname
- 样式文件夹：css
- 脚本文件夹：js
- 样式类图片文件夹：img

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间：2018-06-04
02:34:32

title: 图片命名

命名顺序

图片命名建议以以下顺序命名：

图片业务（可选） + (mod_) 图片功能类别（必选） + 图片模块名称（可选） + 图片精度（可选）

- 图片业务：

- pp_：拍拍
- wx_：微信
- sq_：手Q
- jd_：京东商城
- ...

- 图片功能类别：

- mod_：是否公共，可选
- icon：模块类固化的图标
- logo：LOGO类
- spr：单页面各种元素合并集合
- btn：按钮
- bg：可平铺或者大背景
- ...

- 图片模块名称：

- goodslist：商品列表
- goodsinfo：商品信息
- userava tar：用户头像
- ...

- 图片精度：

- 普清：@1x
- Retina：@2x | @3x
- ...

如下面例子：

公共模块：

wx_mod_btn_goodlist@2x.png
wx_mod_btn_goodlist.png
mod_btn_goodlist.png

非公共模块：

wx_btn_goodlist@2x.png
wx_btn_goodlist.png

btn_goodlist.png

交叉业务协作

业务交叉协作的时候，为了避免图片命名冲突，建议图片名加上业务和模块前缀，如拍拍侧和手Q侧的业务交叉合作时，侧栏导航icon雪碧图命名：

推荐：

pp_icon_mod_sidenav.png

不推荐：

icon_mod_sidenav.png

处理高清图片的时候，命名应该加上图片相应的精度说明

推荐：

jdc_logo@1x.png

jdc_logo@2x.png

不推荐：

jdc_logo.png

jdc_logo_retina.png

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间： 2018-06-04
02:34:32

title: HTML/CSS文件命名

确保文件命名总是以字母开头而不是数字，且字母一律小写，以下划线连接且不带其他标点符号，如：

```
<!-- HTML -->
jdc.html
jdc_list.html
jdc_detail.html

<!-- SASS -->
jdc.scss
jdc_list.scss
jdc_detail.scss
```

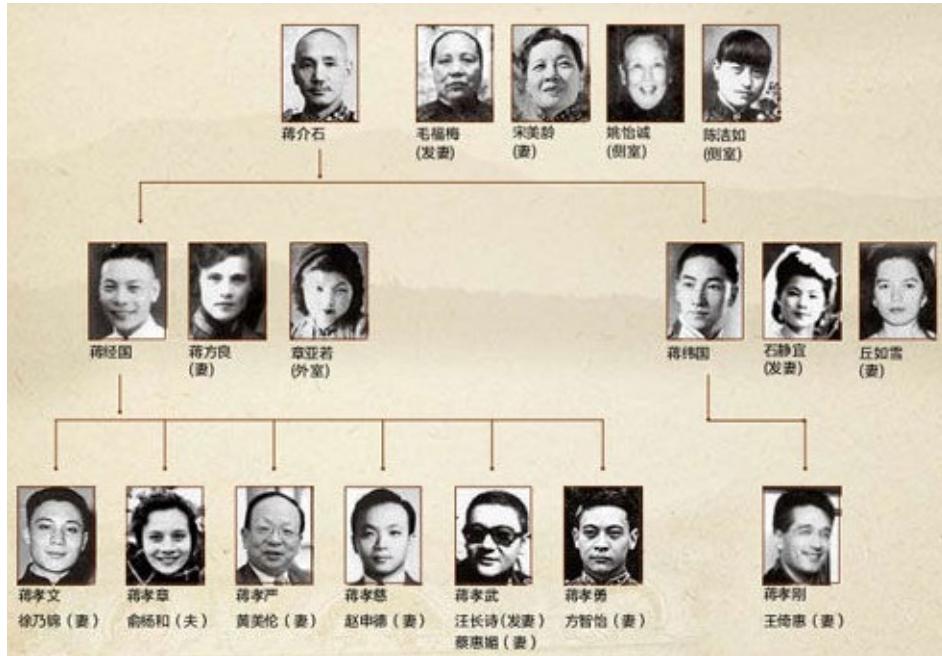
Copyright © stavtop.club 2018 all right reserved, powered by Gitbook该文件修订时间：2018-06-04
02:34:32

title: ClassName命名

ClassName的命名应该尽量精短、明确，必须以字母开头命名，且全部字母为小写，单词之间统一使用下划线“_”连接

命名原则

基于姓氏命名法（继承 + 外来），如下图：



祖先模块不能出现下划线，除了是全站公用模块，如 mod_ 系列的命名：

推荐：

```

<div class="modulename">
    <div class="modulename_info">
        <div class="modulename_son"></div>
        <div class="modulename_son"></div>
        ...
    </div>
</div>

<!-- 这个是全站公用模块，祖先模块允许直接出现下划线 -->
<div class="mod_info">
    <div class="mod_info_son"></div>
    <div class="mod_info_son"></div>
    ...
</div>

```

不推荐：

```
<div class="modulename_info">
    <div class="modulename_info_son"></div>
    <div class="modulename_info_son"></div>
    ...
</div>
```

在子孙模块数量可预测的情况下，严格继承祖先模块的命名前缀

```
<div class="modulename">
    <div class="modulename_cover"></div>
    <div class="modulename_info"></div>
</div>
```

当子孙模块超过4级或以上的时候，可以考虑在祖先模块内具有识别性的独立缩写作为新的子孙模块

推荐：

```
<div class="modulename">
    <div class="modulename_cover"></div>
    <div class="modulename_info">
        <div class="modulename_info_user">
            <div class="modulename_info_user_img">
                <img src="" alt="">
                <!-- 这个时候 miui 为 modulename_info_user_img 首字母缩写-->
                <div class="miui_tit"></div>
                <div class="miui_txt"></div>
                ...
            </div>
        </div>
        <div class="modulename_info_list"></div>
    </div>
</div>
```

不推荐：

```
<div class="modulename">
    <div class="modulename_cover"></div>
    <div class="modulename_info">
        <div class="modulename_info_user">
            <div class="modulename_info_user_img">
                <img src="" alt="">
                <div class="modulename_info_user_img_tit"></div>
                <div class="modulename_info_user_img_txt"></div>
                ...
            </div>
        </div>
        <div class="modulename_info_list"></div>
    </div>
```

```
</div>
```

模块命名

全站公共模块：以 `mod_` 开头

```
<div class="mod_yours"></div>
```

业务公共模块：以 `业务名_mod_` 开头

```
<div class="paipai_mod_yours"></div>
```

常用命名推荐

注意：ad、banner、gg、guanggao 等有机会和广告挂勾的字眼不建议直接用来做ClassName，因为有些浏览器插件（Chrome的广告拦截插件等）会直接过滤这些类名，因此

```
<div class="ad"></div>
```

这种广告的英文或拼音类名不应该出现

另外，**敏感不和谐字眼**也不应该出现，如：

```
<div class="fuck"></div>
<div class="jer"></div>
<div class="sm"></div>
<div class="gcd"></div>
<div class="ass"></div>
<div class="KMT"></div>

...
```

ClassName	含义
about	关于
account	账户
arrow	箭头图标
article	文章
aside	边栏
audio	音频
avatar	头像
bg,background	背景

bar	栏（工具类）
branding	品牌化
crumb,breadcrumbs	面包屑
btn,button	按钮
caption	标题, 说明
category	分类
chart	图表
clearfix	清除浮动
close	关闭
col,column	列
comment	评论
community	社区
container	容器
content	内容
copyright	版权
current	当前态, 选中态
default	默认
description	描述
details	细节
disabled	不可用
entry	文章, 博文
error	错误
even	偶数, 常用于多行列表或表格中
fail	失败 (提示)
feature	专题
fewer	收起
field	用于表单的输入区域
figure	图
filter	筛选
first	第一个, 常用于列表中
footer	页脚
forum	论坛
gallery	画廊

group	模块, 清除浮动
header	页头
help	帮助
hide	隐藏
highlight	高亮
home	主页
icon	图标
info,information	信息
last	最后一个, 常用于列表中
links	链接
login	登录
logout	退出
logo	标志
main	主体
menu	菜单
meta	作者、更新时间等信息栏, 一般位于标题之下
module	模块
more	更多 (展开)
msg,message	消息
nav,navigation	导航
next	下一页
nub	小块
odd	奇数, 常用于多行列表或表格中
off	鼠标离开
on	鼠标移过
output	输出
pagination	分页
pop,popup	弹窗
preview	预览
previous	上一页
primary	主要
progress	进度条
promotion	促销

rcommmd,recommendations	推荐
reg,register	注册
save	保存
search	搜索
secondary	次要
section	区块
selected	已选
share	分享
show	显示
sidebar	边栏, 侧栏
slide	幻灯片, 图片切换
sort	排序
sub	次级的, 子级的
submit	提交
subscribe	订阅
subtitle	副标题
success	成功 (提示)
summary	摘要
tab	标签页
table	表格
txt,text	文本
thumbnail	缩略图
time	时间
tips	提示
title	标题
video	视频
wrap	容器, 包, 一般用于最外层
wrapper	容器, 包, 一般用于最外层

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间: 2018-06-04
02:34:32

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

```
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(declareButNotAssigned) // => throws a ReferenceError
  console.log(typeof declareButNotAssigned) // => 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`

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](#)

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！

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook
该文件修订时间：2018-11-07
17:56:08

title: 语言规范

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 凸凸实验室' +
 '凹凸实验室 凸凸实验室'
```

- 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

```
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(declareButNotAssigned) // => throws a ReferenceError
 console.log(typeof declareButNotAssigned) // => 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`

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间：2018-06-04  
02:34:32

## title: 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坏，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写，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](#)

## 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！

Copyright © stavtop.club 2018 all right reserved, powered by Gitbook 该文件修订时间：2018-06-04 02:34:32