**HTML 部分**

**元素及标签的闭合**

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/>

**HTML 代码大小写**

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**

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>

**特殊字符引用**

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>

**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>

**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>

**多行注释**

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， 表示模块开始， 表示模块结束，模块与模块之间相隔一行

推荐：

<!-- 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：

<!-- 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**

<!-- 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 -->

**CSS 部分**

**样式表编码**

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{}

**代码格式化**

统一使用展开格式

推荐：

.jdc {

display: block;

width: 50px;

}

不推荐：

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

**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{}

**分号**

每个属性声明末尾都要加分号；

**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
* 逗号分隔的取值，逗号之后一个空格
* 为单个css选择器或新申明开启新行
* rgb() rgba() hsl() hsla() rect() 中不需有空格，且取值不要带有不必要的 0
* 属性值十六进制数值能用简写的尽量用简写
* 不要为 0 指明单位

推荐：

.jdc {

width: 100%;

}

.jdc {

box-shadow: 1px 1px 1px #333, 2px 2px 2px #ccc;

}

.jdc,

.jdc\_logo,

.jdc\_hd {

color: #ff0;

}

.nav{

color: #fff;

}

.jdc {

color: rgba(255,255,255,.5);

}

.jdc {

color: #fff;

}

.jdc {

margin: 0 10px;

}

不推荐：

.jdc{

width:100%;

}

.jdc {

box-shadow: 1px 1px 1px #333,2px 2px 2px #ccc;

}

.jdc,jdc\_logo,.jdc\_hd {

color: #ff0;

}.nav{

color: #fff;

}

.jdc {

color: rgba( 255, 255, 255, 0.5 );

}

.jdc {

color: #ffffff;

}

.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";

}

**浏览器私有前缀写法**

浏览器私有前缀在前，标准前缀(无前缀)在后

.jdc {

-webkit-border-radius: 10px;

-moz-border-radius: 10px;

-o-border-radius: 10px;

-ms-border-radius: 10px;

border-radius: 10px;

}

**注释规范**

注释以字符 /\* 开始，以字符 \*/ 结束  
注释不能嵌套

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

多行注释  
注释内容第一个字符和最后一个字符都是一个空格字符，/\* 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：

/\* Comment Text \*/

.jdc {}

/\* Module A

---------------------------------------------------------------- \*/

.mod\_a {}

/\* Module B

---------------------------------------------------------------- \*/

.mod\_b {}

不推荐：

/\*Comment Text\*/

.jdc {}

/\* Module A ---------------------------------------------------- \*/

.mod\_a {}

/\* Module B ---------------------------------------------------- \*/

.mod\_b {}

**文件信息注释**

在样式文件编码声明 [**@charset**](https://github.com/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

\*/

**命名规范部分**

**图片命名**

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**](https://github.com/1x)
  + Retina：[**@2x**](https://github.com/2x) | [**@3X**](https://github.com/3X)
  + …

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

**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

**CSS 类名**

html 的 class 属性本意是作为 html 标签语义的细化和补充

* 尽量少使用样式类
* 使用业务前缀和模块前缀避免命名冲突

推荐：

.sidebar {

float: left;

}

不推荐：

.left {

float: left;

}

具体你使用 BEM 还是其他方法无所谓，清晰即可

**JS Hook**

通常我们使用ID来获取DOM的引用：

document.getElementById('someId')

// or you use jquery

$('#someId')

但是当你需要一次性处理多个元素时，你不能使用 ID，因为ID只能存在于唯一的一个元素上。  
这个时候，用一个特殊的 className 或许是个不错的选择，我们可以规定 js- 前缀的类名是与样式无关的，专供 JS 使用的。  
例如，你可以使用事件代理来处理多个元素的某个类型的事件：

document.body.addEventListenr('click', function (event) {

let matches = [...document.querySelectorAll('.js-some-hook')]

for (let match of matches) {

if (match.contains(event.target)) {

// do your things

break

}

}

})

// or you use jquery

$('body').on('click', '.js-some-hook', function () {

// do your things

})

**JS 部分**

**语法规范**

* 使用 ES6+ 语法时，先去 can i use 网站上查看兼容性，如果兼容性不能覆盖到对应业务场景的 95% 就需要用工具进行编译或放弃这种写法
* 变异数组时，尽量使用 map filter reduce 方法，减少手动遍历，这样可以增强代码的可读性
* 一个函数尽量只完成一项功能，将大的代码块切分成小的函数有利于代码的复用、调试和测试
* 多数情况下分号可以省略，但在一个文件内要保持一致

// bad

const test = 'good';

(function () {

const str = 'hahaha';

})()

// good

const test = 'good'

;(() => {

const str = 'hahaha'

})();

**代码规范**

**单行代码块**

在单行代码块中使用空格

推荐：

function foo () { return true }

if (foo) { bar = 0 }

不推荐：

function foo () {return true}

if (foo) {bar = 0}

**大括号风格**

if (foo) {

bar()

} else {

baz()

}

**变量命名**

使用驼峰式命名

**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

**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' ]

**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。

推荐：

function func () {

// do something

}

// 此处是新的一行

**函数调用**

在函数调用时，禁止使用空格

不推荐：

fn ()

fn

()

**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

推荐：

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

不推荐：

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

**构造函数首字母大写**

**构造函数的括号不可省略**

**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

**禁止使用链式赋值**

不推荐：

var a = b = c = 1

**变量声明**

推荐：

var a

var b

var c

不推荐：

var a, b, c

**代码块空格**

推荐：

if (a) {

b()

}

function a () {}

不推荐：

if (a){

b()

}

function a (){}

**函数声明的空格**

命名函数要求函数名和 function 关键字之间有空格。

推荐：

function func (x) {

// ...

}

不推荐：

function func(x) {

// ...

}

**操作符的空格**

约定操作符前后都需要添加空格

推荐：

var sum = 1 + 2

不推荐：

var sum = 1+2

**代码缩进**

关于代码缩进，是个亘古不变的话题，这里约定使用空格进行缩进（当然可以使用编辑器或者IDE提供的tab转空格的功能）。至于是使用两个空格还是四个空格，只要保持在一个文件内一致就好了。

以下是开源社区较为流行的代码缩进风格：

* .html 4 spaces
* .css 4 spaces
* .js 2 spaces

**一些问题**

**关于目录的划分**

目前很多页面的 js 和 html 是放在同一目录（html/）下的，这样有如下一些缺点：

* 实际文件类型与目录名不匹配，明明是 html 目录，为什么要放 js 文件
* 很容易使得文件数量变得很多，不利于查找文件等

**关于ht.js**

这个文件目前包含的东西很多：常量、工具方法、URL前缀。  
应该把不常变化的东西和经常变化的东西隔离开，把像URL前缀这样需要经常变化的东西单独写成一个配置文件。  
所以建议把ht.js拆成两个文件，一个（比如叫 ht\_prefix.js）专门用来存放接口地址前缀，剩下的常量和工具方法放到另一个文件（比如叫 ht\_common.js）。  
这样在一开始提交的时候，就不要提交 ht\_prefix.js 这个文件了，不然其他人更新下来就会发生冲突。

**颜色和字体大小**

一个好的系统设计不应该随意地去使用颜色和不同的字体大小，不然整个系统会显得乱或者不正式。  
建议设计在设计开始的时候，将要使用的颜色、字体大小收集记录起来，每使用一种新的颜色或者字体大小就记录在同一个地方。  
这样做的好处是，前端可以清楚的知道有那几种颜色和字体大小，并使用自定义变量（ 可能需要postcss sass 等工具处理）来处理，这样一来，修改颜色来变更主题就轻松多了。

**IOS 的常见问题**

**Click 事件代理失效**

当同时满足以下条件时，在 iOS 上 Click 事件代理会失效：

* 元素不是 a[href], input, button 等天生具有可点击含义的元素
* 代理元素是 document.documentElement 或 document.body
* 元素没有应用 cursor: pointer 样式

解决办法也很简单，对应采取措施即可：

* 将元素类型改成可点击的
* 改为在 body 的后代元素上进行事件代理
* 在触发事件的元素上或者其祖先元素一直到代理元素之间的任何元素上添加 cursor: pointer

**margin-bottom 失效**

页面底部元素的 margin-bottom 在 iOS 中失效。

解决办法：

* 使用包裹元素的 padding-bottom 属性
* 使用空白的 div 撑起高度

**其他一些常见错误**

* document.querySelectorAll() 在一些环境下没有 forEach 方法，建议通过 Array.prototype.call() or Array.from() 等方法转成数组再操作。