**XXXX有限公司**

**Web前端代码编写规范**

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ √ ] 草稿  [ ] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本： | 1.0 |
| 作 者： |  |
| 完成日期： |  |

2018年6月**修改历史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **修订后**  **版本号** | **修订人** | **修订内容简述** | **修订**  **日期** |
| 1 | 1.0 |  | 起草 | 2018.6.11 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 第一部分 HTML规范

HTML并不是一种编程语言，而是一种标记语言，它没有任何真正的编程语言中的循环或是流程控制语句。然而，HTML代码的格式和风格是非常重要的，因为要经常对HTML代码进行维护和修改，因此HTML代码必须有很清晰的逻辑结构和布局，而使其易懂和易于维护。

HTML 语言是不区分大小写的，但为改善可读性，规定小写所有字母（除最上一行引用外）。与HTML不一样，XHTML对大小写是敏感的，<title>和<TITLE>是不同的标签。

具体规范如下：

## 1.文件定义

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

## 2.网站head区代码规范：

head区是指<head>和</head>之间的内容。必须加入的标签：

### 2.1公司版权注释

<!---

文件：XXX.html

说明：说明本HTML是用于什么功能

作者：李四

时间：2016/1/30

--->

### 2.2网页显示字符集

**简体中文**：<meta http-equiv="Content-Type" content="text/html; charset=**utf-8**">

繁体中文：<meta http-equiv="Content-Type" content="text/html; charset=**BIG5**">

英 语：<meta http-equiv="Content-Type" content="text/html; charset=**iso-8859-1**">

### 2.3作者与版权信息

此处用于存储开发公司版权信息，如：

<meta name="author" content="www.hezo.cn">

<meta name="copyright" content="www.hezo.cn,版权所有"/>

### 2.4网页的css,javascript规范

在引用CSS和JS文件时，统一放置在<head></head>中，先放CSS引用，再放js引用。

<link href="style/style.css" rel="stylesheet" type="text/css" />

html中javascript的书写：

<script language="javascript" type="text/javascript"></script>

### 2.5网页标题

Title中使用“-”分割，如：设备管理平台-运维系统。

### 2.6设置搜索关键词

<meta name="keywords" content=" " />

百度关键词搜索指数：http://index.baidu.com/，在这个工具里，可以看到一些常用关键词的每日搜索次数。

热门词汇，目前收索量最多的词汇。可以查看百度热词榜，http://top.baidu.com/index2.html

Google AdWords关键字工具：查询特定关键词的常见查询及扩展匹配。

KEYWRODS 关键字最多不超过5个建议长度：小于等于100个汉字，如果使用的网页编码是UTF-8，不同关键字之间用英文“,”分隔。

### 2.7网站简介

<meta name="description" content=" " />

description是描述网页内容的，因此最好是可以用一句话来概括本网页的主题内容。

description不要超过255个字符，搜索引擎索引一般都会索引Description的前255个字符，因此，这255个字符是做搜索引擎优化的关键。

如果使用的网页编码是UTF-8，那么在Description中的标点符号最好都使用英文，例如英文的逗号、句号等。如果是中文的话，那么就使用中文的标点符号即可。

### 2.8其它标签[非必需]

1.设定网页的到期时间。一旦网页过期，必须到服务器上重新调阅，这样有利于页面信息。

<meta http-equiv="expires" content="wed, 26 feb 2009 08:21:57 gmt">

2.禁止浏览器从本地机的缓存中调阅页面内容。

<meta http-equiv="pragma" content="no-cache">

3.用来防止别人在框架里调用你的页面。

<meta http-equiv="window-target" content="\_top">

4.自动跳转。

<meta http-equiv="refresh" content="5;url=http://www.abc.cn"> 5指时间停留5秒。

5.网页搜索机器人向导.用来告诉搜索机器人哪些页面需要索引，哪些页面不需要索引。

<meta name="robots" content="none">

CONTENT的参数有all,none,index,noindex,follow,nofollow。默认是all。

6.收藏夹图标

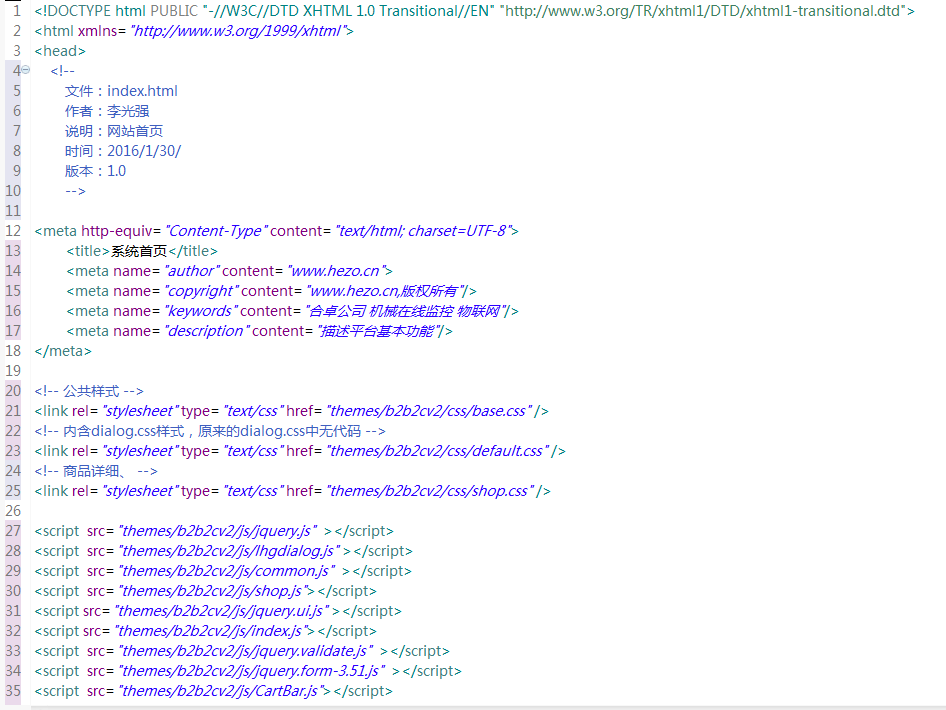
<link rel="shortcut icon" href="/favicon.ico" type="image/x-icon"/> 图片大小16px\*16px

7.订阅RSS浏览

是一种描述和同步网站内容的格式。用户可以通过RSS阅读器订阅。

<link href=”abc.aspx” title=”新现代教育网” type=”application/rss+xml” rel=”alternte” />

### 2.9完整示例



## 3. html中元素的书写：

**<body>区域标签**

### 3.1段落

段落使用 <p> 标签进行定义。

<p>This is a paragraph</p>

### 3.2标题

即正文标题一般用<h1></h1>标签，最好和title标签的内容有关联。

标题使用 <h1> 至 <h6> 标签进行定义。<h1> 定义最大的标题。<h6> 定义最小的标题。

<h1>This is a heading</h1>

<h2>This is a heading</h2>

<h3>This is a heading</h3>

<h4>This is a heading</h4>

<h5>This is a heading</h5>

<h6>This is a heading</h6>

文章中的小标题一定也要加粗。

可以把网页中的重要内容做成这样。用户不用登陆这个网页就可看到更新的内容。

### 3.3字体的设置标签

换行：<br/>

字体修饰不要使用<font><strong>等进行标注，一般使用css样式进行字体样式排版

### 3.4 注意事项

**1.<body>为了保证浏览器的兼容性**，必须设置页面背景：

<body bgcolor="#FFFFFF">

2.关闭所有的标签

打开的标签必须关闭，例如<p>www.abc.cn</p>，当然还有一种关闭方式，如：<br/>。即：一定要有结束标记。

**3. 属性值用双引号** “” 括起来,属性统一使用小写

例如：<img height="60" width="760" src="images/logo.gif"/>

**4. 给所有的属性赋值**

不正确的写法：<input …… checked />

正确的写法为：<input …… checked= "checked"/>

**5.标签要合理嵌套**

不正确的写法：<div><h1> www.abc.cn</div></h1>

正确的写法为：<div><h1>www.abc.cn</h1></div>

**7.特殊字符用编码标识**

如 " <" 用 "&lt;"表示，" >" 用 "&gt;"表示。

**8.网页中图片优化**

一定要加alt属性，alt属性指定了当图片不能显示的时候就显示供替换文本。 内容最好与关键词相关。

如：<img src="images/logo.gif" alt="网站LOGO"/>

一些小图标最好用.png或.gif格式图片，比如说箭头，文章列表前的图标等，在CSS中做成背景。这些小图标最好做一张图片，用CSS来定位。

**9.用结构化的元素输出内容**

<ul>

<li>新现代</li>

<li>新现代</li>

<li>新现代</li>

</ul>

或者使用

<dl>

<dt></dt>

<dd></dd>

</dl>

10．**标签要有缩进，便于代码阅读修改**。

**11、 tr、td必须定义在table之间**；

<table>

<tbody>

<tr>

<td>……</td>

….

</tr>

</tbody>

</table>

注：**tbody绝对不可以省略**。

1**2、 button按钮必须定义在form之间**，否则netscape不支持；

<form action=”……” method=”post”>

<input type=”button” name=”but” value=”back”>

</form>

**13、html注释**

(1) 每一独立部分都要表明注释。例如：

<!—右边 begin--> <!--右边end-->

<!—左边 begin--> <!—左边end-->

**块代码的注释**

<!—头部 begin-->

大量代码块，使用此种样式，方便阅读

<!—头部end-->

(2) 注释要用描述性语言，能够清楚的表达每一部分的意思。

## 4. HTML元素的排序规则

能够使用的HTML元素包括：div、p、ul、table、span、input、select等。基本上div、table、ul、p都属于结构性比较强的元素，而span、input则是比较弱的元素，因此不允许有span嵌套div、table等等的情况出现，span可以嵌套input, 可以嵌套span。

在写页面的时候可以先不考虑界面呈现，按照这样的规则，把数据直接绑定到HTML元素的节点上。

## 5.九大原则

**原则一**：

一般要写兼容多种浏览器的网页，最省事的方法就先写适合chrome的样式，并在chrome进行调度。测试时可以使用多种浏览器。

**原则二**：

逐步叠加的方式，如果需要用这个样式的模块很多，并且会存在些许的不同，千万不要把样式一步写到位，最好把框架性的先写好，细节性的留到第二个步骤，甚至第三个步骤，当然最好步骤不要超过三步，这样很不利于维护。

由若干个二级定义对一级定义进行补充，这样的既可以减少代码，又可以减少繁多的CLASS命名。

不要轻易修改共用的CSS，否则可能会影响其它页面。

在每个一级CSS前添加注释，说明哪些页面使用了该CSS。

**原则三**：

从总到分。顺序应该是这样：总体需要用到的>>结构>>模块>>二级模块>>细节；并且这个文档最不容易修改的处于最上端，频繁用于修改的在末端。最忌讳即兴发挥，这样写出的东西经不起推敲，稍有变化就要忙得乱成一团。结构性的东西应该慎重精细。

**原则四**：

尽量把能够放在一组的小图标或者图片放在一个图片文件里，这样的好处有很多。

1、 图片文件的总量会变小；

2、利于下载，太多的小图片下载效果是不理想的，常常会发现这个图片出来，那个没出来，这样在写样式的时候只需要写一个总体的，对于具体的要显示的只需要标一个位置，非常省时间，样式也很精炼；

3、 显而易见，这样做非常便于管理，替换或新增的时候工作量很小。

**原则五**：

切图的原则，应该切大的图绝不切小，应该切小的绝不切大，能够用颜色代替的绝不切图。把一个大图切成很多份并不一定能够加快页面的显示速度，相反会浪费很多不必要的带宽。不光要控制图片文件的多少，还要考虑到这样切会不会造成页面增加许多额外的代码。

**原则六**：

图片使用能名称表达图片意义的名词命名，尽量不要使用编号命名。

**原则七**：

涉及多行多列的特别是列宽有不固定的，坚决用表格，千万不要为了DIV而DIV。

多个图片规则排列时，可以使用<dl><dt></dt><dd></dd></dl>。

**原则八**：

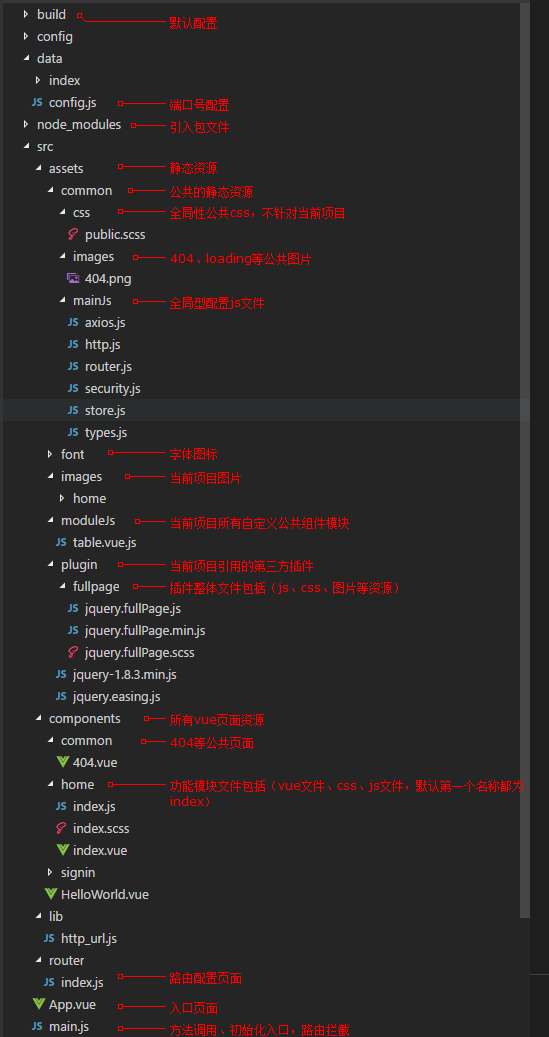
尽可能不把样式直接写到元素的标签里（*除非这个样式出现的几率特别小，有相当程度的偶然性*）。

**原则九**：

尽量不要用ID来定义或索引样式，ID的唯一性比较高，且用ID索引样式效率非常低而且兼容性的问题也比较明显。

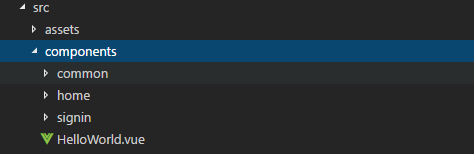
## 6．文件命名原则

### 6.1 文件目录结构



### 6.2 前台文件组织

在前台页面文件中，每个大类功能模块（子系统）单独创建一个文件夹，使用与之相关的有意义的英文名词命名。例如有一个系统首页（home）、登陆（signin）等模块，文件夹组织：

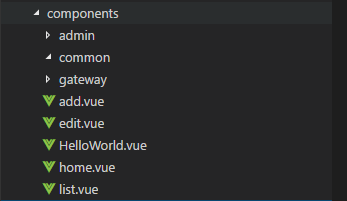


每个功能模块中应该包含一个缺省的html或vue 文件，文件名统一用index.html，即当用户缺省访问该目录时，自动加载该页面。

公用页面（如登录404.html、login.html、页头header.html、页脚footer.html、导航navigation.html、菜单menu.html、信息/错误提示message.html等），可以统一放在common目录中。

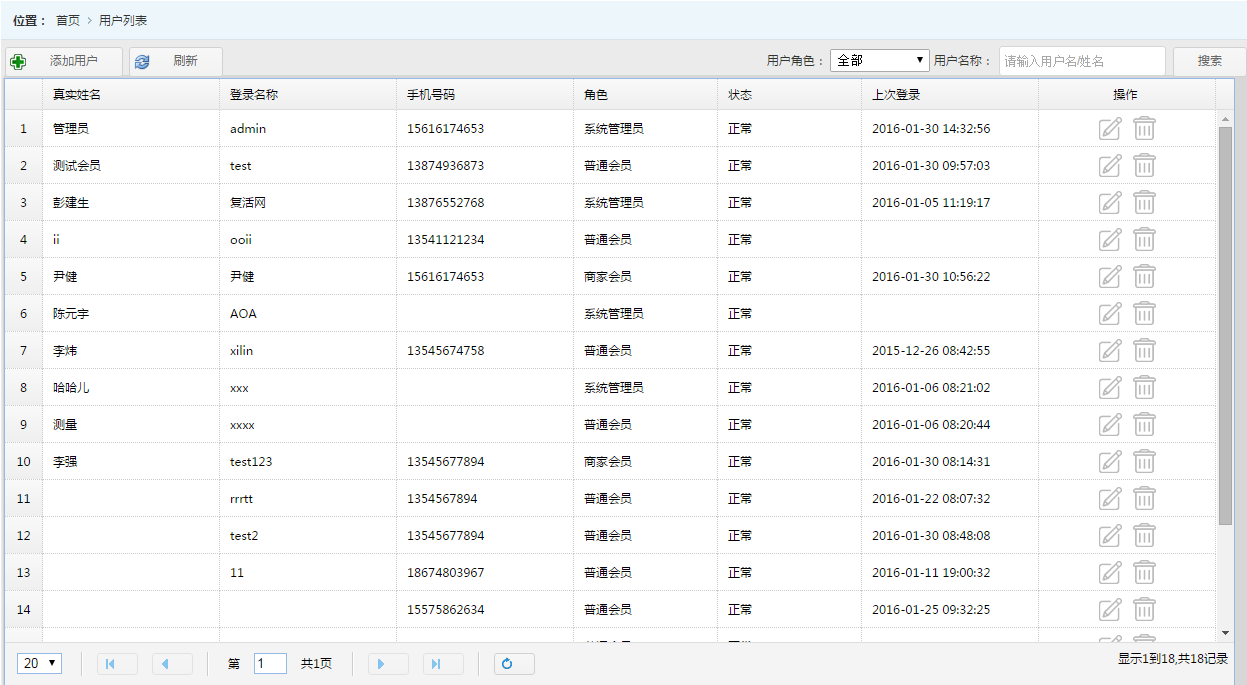
### 6.3 页面命名规则

具有管理功能的页面，如用名、角色、设备、日志、文章等的管理，可以归纳为列表（list.htm）、新建（add.html）、编辑（edit.html）等几类操作，所以在页面组织时，每类模块单独创建一个文件夹，使用能够表达模块意义的英文名词命名，并在其中分别创建list.html、add.html、edit.html等文件，对应于内容列表、添加、编辑等操作。示例：



**其它说明：**

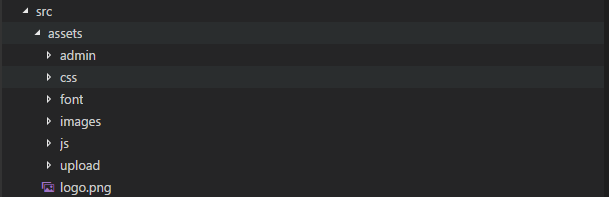
1. 在列表页面里统一使用“四栏式”显示风格，即列表页面里必须包括模块目录/位置提示、工具栏（含添加、编辑、删除、刷新、查询等操作控件）、表格、分页操作等部分。示例：



1. 表格用于显示查询内容，在list.html打开时，必须默认加载一定的记录；表格栏目仅显示用户常用或关心的字段；在显示时必须对表格内容进行必要的格式处理；对记录常用的操作可以放置于操作栏内（如上图中的操作栏内放置有编辑和删除两个操作按钮）。
2. 表格可以统一使用easyui datagrid或bootstap data tables。

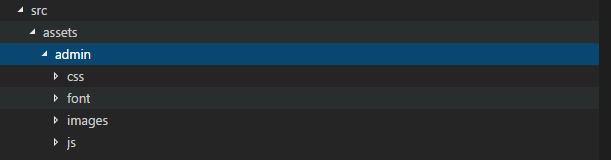
### 6.4 资源文件的命名原则

资源文件包括css、js、图片等内容。所有资源文件统一存储在web根目录assets文件夹中。示例：



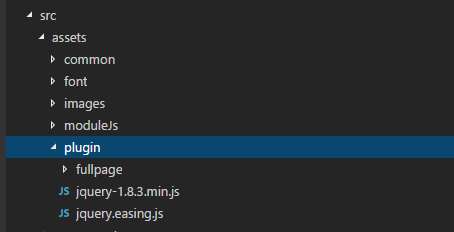
每个子系统（大类）文件夹命名规则同6.1规则。

每个子系统文件夹中分别包括css、js和images，示例：



多个子系统公共使用的资源，统一存储在common文件夹中，如常用的js操作，可以命名为generic.js，放置在common/js/generic.js。

### 6.5外引入插件等资源存放位置

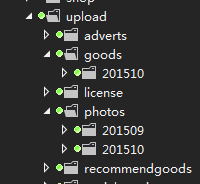


### 6.6 图片命名规则

图片分成两部分，一是网页设计时使用的静态资源，通常不会因系统的运行而发生变化，另一类是程序生成、用户上传、终端上传的图片。

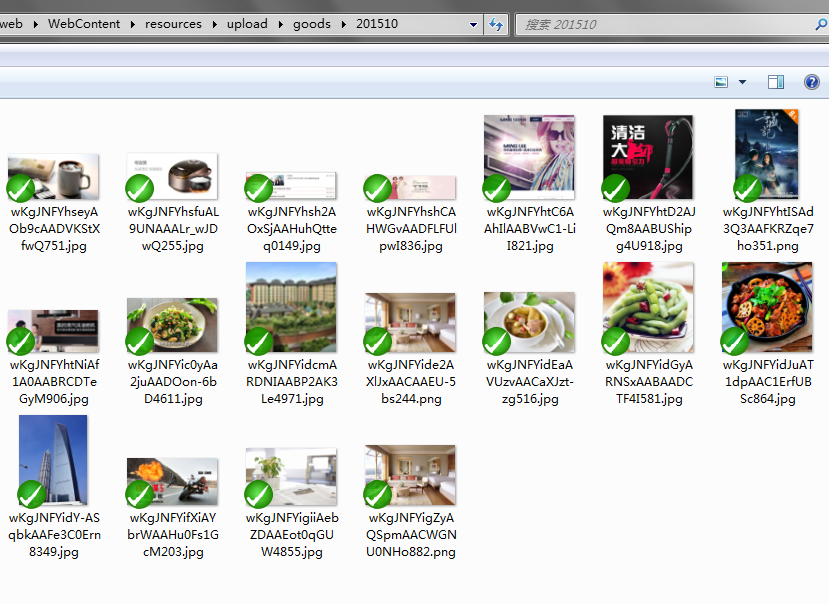
（1）静态图片存储在assets\大类目录\images中，使用能表达图片意义的英文单词命名，不能使用图片编号命名。网面使用的小图标，可以放置在一个图片文件中，使用css裁切和背景方式显示在页面中。

（2）动态图片存储在assets\upload目录中，在其中创建分类图片目录。例如，与系统相关的图片存储在system目录中，设备监控相关图片存储在management中等。为了避免同一文件夹中文件数量限制，在分类目录中，按上传日期分成不同的次级目录，命名规则为上传文件所在的yyyyMM（年度月份），其中月份使用两位数字表示。例如：



上传的图片统一使用GUID标识进行命名，即在上传时，由接收的java程序自动生成guid编号+.图片扩展名进行命名。

**注意：**保存图片时，文件扩展名必须保持不变。示例：



## 7.常用HTML标识符

**HTML标记一览:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 标记 | 类型 | 译名或意义 | 作 用 | 备注 |
| 文件标记 | | | | |
| <HTML> | ● | 文件声明 | 让浏览器知道这是 HTML 文件 |  |
| <HEAD> | ● | 开头 | 提供文件整体资讯 |  |
| <TITLE> | ● | 标题 | 定义文件标题，将显示于浏览顶端 |  |
| <BODY> | ● | 本文 | 设计文件格式及内文所在 |  |
| 排版标记 | | | | |
| <!--注解--> | ○ | 说明标记 | 为文件加上说明，但不被显示 |  |
| <P> | ○ | 段落标记 | 为字、画、表格等之间留一空白行 |  |
| <BR> | ○ | 换行标记 | 令字、画、表格等显示于下一行 |  |
| <HR> | ○ | 水平线 | 插入一条水平线 |  |
| <CENTER> | ● | 居中 | 令字、画、表格等显示于中间 | 反对 |
| <PRE> | ● | 预设格式 | 令文件按照原始码的排列方式显示 |  |
| <DIV> | ● | 区隔标记 | 设定字、画、表格等的摆放位置 |  |
| <NOBR> | ● | 不折行 | 令文字不因太长而绕行 |  |
| <WBR> | ● | 建议折行 | 预设折行部位 |  |
| 字体标记 | | | | |
| <STRONG> | ● | 加重语气 | 产生字体加粗 BOLD 的效果 |  |
| <B> | ● | 粗体标记 | 产生字体加粗的效果 |  |
| <EM> | ● | 强调标记 | 字体出现斜体效果 |  |
| <I> | ● | 斜体标记 | 字体出现斜体效果 |  |
| <TT> | ● | 打字字体 | COURIER字体，字母宽度相同 |  |
| <U> | ● | 加上底线 | 加上底线 | 反对 |
| <H1> | ● | 一级标题标记 | 变粗变大加宽，程度与级数反比 |  |
| <H2> | ● | 二级标题标记 | 将字体变粗变大加宽 |  |
| <H3> | ● | 三级标题标记 | 将字体变粗变大加宽 |  |
| <H4> | ● | 四级标题标记 | 将字体变粗变大加宽 |  |
| <H5> | ● | 五级标题标记 | 将字体变粗变大加宽 |  |
| <H6> | ● | 六级标题标记 | 将字体变粗变大加宽 |  |
| <FONT> | ● | 字形标记 | 设定字形、大小、颜色 | 反对 |
| <BASEFONT> | ○ | 基准字形标记 | 设定所有字形、大小、颜色 | 反对 |
| <BIG> | ● | 字体加大 | 令字体稍为加大 |  |
| <SMALL> | ● | 字体缩细 | 令字体稍为缩细 |  |
| <STRIKE> | ● | 画线删除 | 为字体加一删除线 | 反对 |
| <CODE> | ● | 程式码 | 字体稍为加宽如<TT> |  |
| <KBD> | ● | 键盘字 | 字体稍为加宽，单一空白 |  |
| <SAMP> | ● | 范例 | 字体稍为加宽如<TT> |  |
| <VAR> | ● | 变数 | 斜体效果 |  |
| <CITE> | ● | 传记引述 | 斜体效果 |  |
| <BLOCKQUOTE> | ● | 引述文字区块 | 缩排字体 |  |
| <DFN> | ● | 述语定义 | 斜体效果 |  |
| <ADDRESS> | ● | 地址标记 | 斜体效果 |  |
| <SUB> | ● | 下标字 | 下标字 |  |
| <SUP> | ● | 上标字 | 指数（平方、立方等） |  |
| 清单标记 | | | | |
| <OL> | ● | 顺序清单 | 清单项目将以数字、字母顺序排列 |  |
| <UL> | ● | 无序清单 | 清单项目将以圆点排列 |  |
| <LI> | ○ | 清单项目 | 每一标记标示一项清单项目 |  |
| <MENU> | ● | 选单清单 | 清单项目将以圆点排列，如<UL> | 反对 |
| <DIR> | ● | 目录清单 | 清单项目将以圆点排列，如<UL> | 反对 |
| <DL> | ● | 定义清单 | 清单分两层出现 |  |
| <DT> | ○ | 定义条目 | 标示该项定义的标题 |  |
| <DD> | ○ | 定义内容 | 标示定义内容 |  |
| 表格标记 | | | | |
| <TABLE> | ● | 表格标记 | 设定该表格的各项参数 |  |
| <CAPTION> | ● | 表格标题 | 做成一打通列以填入表格标题 |  |
| <TR> | ● | 表格列 | 设定该表格的列 |  |
| <TD> | ● | 表格栏 | 设定该表格的栏 |  |
| <TH> | ● | 表格标头 | 相等于<TD>，但其内之字体会变粗 |  |
| 表单标记 | | | | |
| <FORM> | ● | 表单标记 | 决定单一表单的运作模式 |  |
| <TEXTAREA> | ● | 文字区块 | 提供文字方盒以输入较大量文字 |  |
| <INPUT> | ○ | 输入标记 | 决定输入形式 |  |
| <SELECT> | ● | 选择标记 | 建立 POP-UP 卷动清单 |  |
| <OPTION> | ○ | 选项 | 每一标记标示一个选项 |  |
| 图形标记 | | | | |
| <IMG> | ○ | 图形标记 | 用以插入图形及设定图形属性 |  |
| 连结标记 | | | | |
| <A> | ● | 连结标记 | 加入连结 |  |
| <BASE> | ○ | 基准标记 | 可将相对 URL 转绝对及指定连结目标 |  |
| 框架标记 | | | | |
| <FRAMESET> | ● | 框架设定 | 设定框架 |  |
| <FRAME> | ○ | 框窗设定 | 设定框窗 |  |
| <IFRAME> | ○ | 页内框架 | 于网页中间插入框架 | IE |
| <NOFRAMES> | ● | 不支援框架 | 设定当浏览器不支援框架时的提示 |  |
| 影像地图 | | | | |
| <MAP> | ● | 影像地图名称 | 设定影像地图名称 |  |
| <AREA> | ○ | 连结区域 | 设定各连结区域 |  |
| 多媒体 | | | | |
| <BGSOUND> | ○ | 背景声音 | 于背景播放声音或音乐 | IE |
| <EMBED> | ○ | 多媒体 | 加入声音、音乐或影像 |  |
| 其他标记 | | | | |
| <MARQUEE> | ● | 走动文字 | 令文字左右走动 | IE |
| <BLINK> | ● | 闪烁文字 | 闪烁文字 | NC |
| <ISINDEX> | ○ | 页内寻找器 | 可输入关键字寻找于该一页 | 反对 |
| <META> | ○ | 开头定义 | 让浏览器知道这是 HTML 文件 |  |
| <LINK> | ○ | 关系定义 | 定义该文件与其他 URL 的关系 |  |
| STYLESHEET | | | | |
| <STYLE> | ● | 样式表 | 控制网页版面 |  |
| <SPAN> | ● | 自订标记 | 独立使用或与样式表同用 |  |

注：

* ● 表示该标记属围堵标记，即需要关闭标记如 </标记>。
* ○ 表示该标记属空标记，即不需要关闭标记。
* IE 表示该标记只适用于 INTERNET EXPLORER。
* NC 表示该标记只适用于 NETSCAPE COMMUNICATOR。
* 反对 表示该标记不为 W3C 所赞同，通常这标记是 IE 或 NC 自订，且己为众所支 持，只是 HTML 标准中有其它同功能或更好的选择。

弃用 表示该标记己为 W3C 所弃用，是过时的标记，但 HTML 具向下兼容的特 性，不用担心新浏览器不支援旧标记。

# 第二部分[CSS编码规范](http://www.cnblogs.com/hide0511/archive/2006/10/09/524304.html)

## ****1. 基本书写规范****

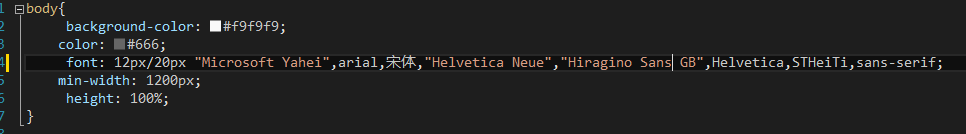
### **1.1所有的CSS的尽量采用外部调用**

　　 <LINK href="/style/style.css" rel="stylesheet" type="text/css">

　　 书写时重定义的最先，伪类其次，自定义最后（其中a:link a:visited a:hover a:actived 要按照顺序写）便于自己和他人阅读。

　　 为了保证不同浏览器上字号保持一致，字号建议用点数pt和像素px来定义，pt一般使用中文宋体的9pt和11pt，px一般使用中文宋体12px 和14.7px 这是经过优化的字号，黑体字或者宋体字加粗时，一般选用11px和14.7px 的字号比较合适。

字体选择：**优先使用微软雅黑，其次为宋体，可以统一在body中定义**，而自定义样式中，不需要重复定义字体。如：



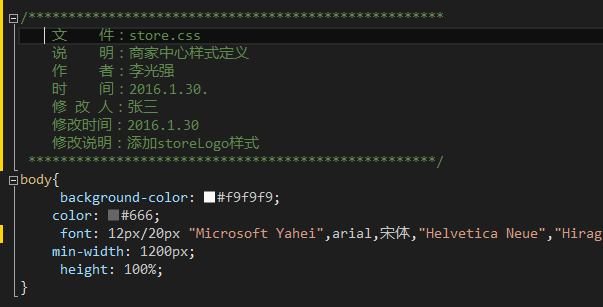
### 1.2 CSS推荐模板

模板  
body {dot}{ font-size:9pt; font-family:Arial, Helvetica, sans-serif; color:#333333; text-align:center; margin:0px;}  
ul {dot}{ margin:auto; }  
img {dot}{ border:0px; }  
a {dot}{ font-size:9pt; text-decoration:none;color:#FFFFFF; }  
a:hover {dot}{ font-size:9pt; text-decoration:underline; color:#990000; }  
a.1 {dot}{ font-size: 9pt; color: #3366cc; text-decoration: none}   
a.1:hover {dot}{ font-size: 9pt; color: #FF9900; text-decoration: none}  
None.colorblue,colorblue:hover {dot}{ color:#003366 ;}.blue {dot}{ font-family: "宋体"; font-size: 9pt; line-height: 20px; color: #0099FF; letter-spacing: 5em}  
None.colorRed,a.colorRed:hover{dot}{ color: #FF0000; }.colorLime,a.colorLime:hover{dot}{ color: #00FF00; }.colorGreen,a.colorGreen:hover{dot}{ color: #008000; }.colorBlue,a.colorBlue:hover{dot}{ color: #0000FF; }.colorOrange,a.colorOrange:hover{dot}{ color: #FFA500; }

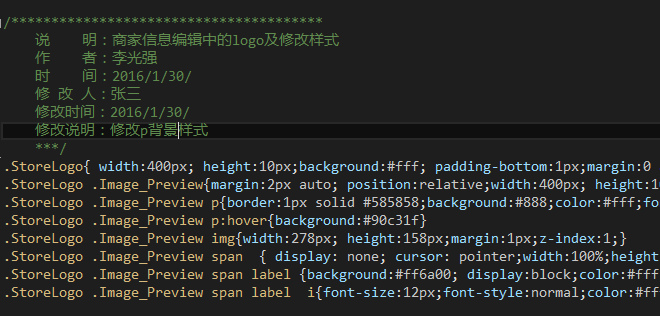
  为了保证浏览器的兼容性，必须设置页面背景<body bgcolor="#FFFFFF">

### 1.3 CSS注释写法

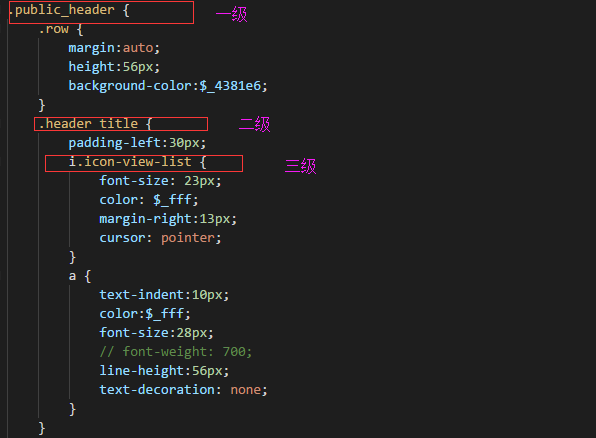
在定义的CSS文件顶部添加文件说明、作者、时间等说明，如例：



一级CSS上面必须加上注释，说明样式的用处。必要时可以添加作者、创建时间、修改人、修改时间。示例：



### 1.4子级元素查找不超过三级的嵌套



### 1.5展示动画效果以css实现为主，js为辅

### 1.6 css兼容性前缀书写采用，使用postCss自动追加

### 1.7 层级包裹，最外层以当前模块或功能区分



### 1.8 不书写行内样式

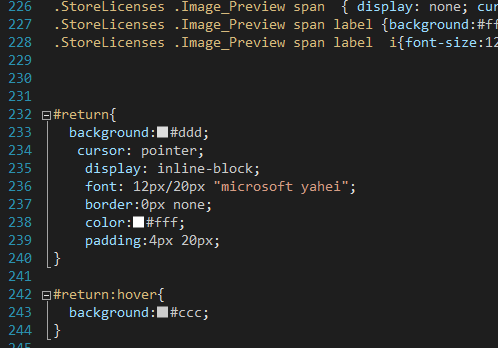
### 1.9 内嵌样式超过50行单独穿件css文件

### 1.10 行内元素不能包含块级元素

## ****2.CSS命名参考****

**CSS文件名统一使用小写英文单词命名。**

**样式定义变量统一使用class命名方式，不建议使用id命名方式**。如下图中，使用“.”定义的样式为class命名方式，使用“#”引导的样式为id命名方式。Id命名方式仅针对于html中使用id定义的元素，通常复用性较差。



### 2.1 常用的CSS命名规则：

头：header

内容：content/container

尾：footer

导航：nav

侧栏：sidebar

容 器：container/box主 导 航：mainNav子 导 航：subNav顶 导 航：topNav网站标志：logo大 广 告：banner页面中部：mainBody菜 单：menu菜单内容：menuContent子 菜 单：subMenu子菜单内容：subMenuContent搜 索：search搜索关键字：keyword搜索范围：range标签文字：tagTitle标签内容：tagContent当前标签：tagCurrent/currentTag

标　 题：title列 表：list当前位置：currentPath图 标：icon注 释：note登 录：login注 册：register栏目：column

列 定 义：column\_1of3 (三列中的第一列)column\_2of3 (三列中的第二列)column\_3of3 (三列中的第三列)

### 2.2直观命名

当在设计Web页面以及需要对一个div进行标识的时候，最自然的想法就是使用可以描述元素所在页面位置的词汇来对其命名。如：

自上而下分组：top\_panel

横向导航：horizontal\_nav

左面边栏：left\_side

中心-栏目：center\_column

右面：right\_col

等下划线进行连接，连接单词不超过三个

### 2.3 CSS复用

不需要重复定义可继承的值，子元素自动继承父元素的属性值，象颜色、字体等，已经在父元素中定义过的，在子元素中可以直接继承，不需要重复定义，除非是为了更变当前元素样式不使用父元素的属性值，但是要注意，浏览器可能用一些默认值覆盖你的定义。

### 2.4.多重CSS样式定义

CSS属性追加重复最后优先原则，即一个标签可以同时定义多个class，也可以是同一个class中重复定义属性。例如： 先定义两个样式.one{width:200px;background:url(http://www.86w3.com/1.jpg) no-repeat left top;}.two{border:10px solid #000; background:url(http://www.86w3.com/2.jpg) no-repeat left top;} 在页面代码中，我们可以这样调用: <div class=one two></div> 当应用两个或多个样式时，浏览器所应用的样式根据是属性追加重复最后优先原则。即两个或多个或重复的样式名定义，浏览器所应用的样式是按先后顺序的，如果定义了重复的属性值，以最后定义的为准，如果应用了两个或多个样式名，里面不重复定义的属性值就追加上去，重复的属性值就以最后一个为准。这里要注意的是，样式的先后不是根据页面上应用的名字顺序，而是样式表里的样式顺序。

### 2.5.CSS作用域

CSS的作用域是通过书写方式来实现的更准确的说法应该叫做选择器如果你深入了解后，同样能达到编程思想中的“命名空间”、“全局”、“私有”、“继承”等“编程特性”更能体现他特点的也就是他名字中的“层叠样式”。

如下面的p的作用域：

/\*作用域：全局\*/ p{text-indent:2em;}

/\*作用域：.demo这个类中\*/

.demo p{color:#000000;}

## 3. CSS样式书写顺序

### 3.1 显示属性

\* display \* list-style \* position \* float \* clear

### 3.2自身属性

\* width \* height \* margin \* padding \* border \* background

### 3.3文本属性

\* color \* font \* text-decoration \* text-align   
\* vertical-align \* white-space \* other text \* content

## 4.其它规范

### 4.1 图片样式

禁止用 <img width=? height=?> 来人为干预图片显示的尺寸，即在HTML代码中的 <img> 标签中不要带上width 和height 两个属性，因为制作过程中，图片往往需要反复的修改。

但是使用CSS控制图片大小时，当网页还未加载图片时，不会留出图片的占位大小，可能会造成网页在加载过程中抖动（如果图片是插在一个固定大小的表格里的，不会有这个现象），尤其是当图片的尺寸较大时，这种现象会很明显。所以当预料到这种会明显导致网页抖动的情况会发生时，务必给 <img>附上 width 和 height 属性。

### 4.2换行

为了最大程度的发挥浏览器自动排版的功能，在一段完整的文字中请尽量不要使用<br> 来人工干预分段。

### 4.3 字体

所有的字号都应该用样式表来实现，禁止在页面中出现 <font size=?> 标记。

请不要在网页中连续出现多于一个的也尽量少使用全角空格（英文字符集下，全角空格会变成乱码），空白应该尽量使用 text-indent, padding, margin, hspace, vspace 以及透明的gif 图片来实现。

中英文混排时，我们尽可能的将英文和数字定义为verdana 和arial 两种字体。

行距建议用百分比来定义，常用的两个行距的值是line-height:120%/150%.

# 第三部分 JavaScript规范

## 1.命名规范

### 1.1基本原则

规范的命名能使程序更易阅读，从而更易于理解。它们也可以提供一些标识功能方面的信息，有助于更好的理解代码和应用。

* 使用可以准确说明变量/函数/原型(prototype)的完整英文描述符。例如， firstName，listAllUsers 或 CorporateCustomer等，避免使用汉语拼音、不相关单词及汉字进行命名。
* 采用大小写混合，提高名字的可读性。一般应该采用小写字母，但是原型(prototype)命名时任意单词的首字母大写。
* **尽量少用缩写**，但如果一定要使用，当使用公共缩写和习惯缩写等，如实现（implement）可缩写成impl，经理（manager）可缩写成mgr等，严禁滥用缩写。
* 避免使用长名字（最好不超过 25 个字母）。
* 避免使用相似或者仅在大小写上有区别的名字。
* **避免使用数字，但可用2代替to，用4代替for等**，如：go2Jsp。此外，若元素id包含数字则除外

### 1.2 Prototype/Method命名规范

#### 1.2.1原型Prototype

所有单词首字母大写。使用能确切反应该原型含义、功能等的词。一般采用名词。

#### 1.2.2方法Method/函数Function

方法的命名应采用完整的、能够清楚表达该方法功能的英文描述符，首字母小写，其他单词的首字母大写。方法名称的第一个单词常常采用一个有强烈动作色彩的动词。

取值类使用get前缀，设值类使用set前缀，判断类使用is(has)前缀。如下例所示：

getName()

setSarry()

isLogon()

方法参数建议顺序：(被操作者，操作内容，操作标志，其他……)。如下例所示：

function replace (sourceStr, //源字串

oldStr, //被替换字串

newStr){ //替换为字串

}

### 1.3字段命名规范

#### 1.3.1常量

采用完整的英文大写单词，在词与词之间用下划线连接，如：DEFAULT\_VALUE

#### 1.3.2 变量

变量声明使用let，全局变量声明用const

除第一个单词外其余单词首字母大写。

对私有实例变量可使用\_前缀，但在其存取方法中则应该将其前缀去掉。

#### 1.3.3集合

一个集合，例如数组或xml节点集合，应采用复数命名来表示变量中存放的是一组对象。命名应采用完整的英文描述符，名字中所有非首单词的第一个字母应大写，适当使用集合缩写前缀。如下例所示：

集合变量命名示例

var rowNodes = xmlDom.selectNodes(“/\*/data/row”);

var aryUsers = []; //用户列表

#### 1.3.4特殊值

我们在程序里经常会用到一些量，它是有特定的含义的。例如，现在我们写一个薪金统计程序，公司员工有50人，我们在程序里就会用50这个数去进行各种各样的运算。在这里，50就是“特殊值”。当别的程序员在程序里看到50这个数，将很难知道它的含义，造成理解上的困难。

在程序里出现“特殊值”会降低程序的可读性、可维护性和可扩展性，故规定不得出现此类“特殊值”。**避免的方法是把“特殊值”定义为一个常量**。注意这个常量的命名应该能表达该数的意义，并且应该全部大写，且单词之间使用下划线。例如上面50这个数，我们可以定义一个名为NUM\_OF\_EMPLOYEES的常量。这样，别的程序员在读程序的时候就可以很容易的理解了。

### 1.4异常

必要时，在javascript代码段中加入异常捕获代码。通常情况下不需要自定义异常原型，仅使用内建error对象。

异常原型由表示该异常类型的单词和Exception组成，如ResponseException。

异常实例一般使用e、ex等，在多个异常时使用该异常名或简写加E，Ex等组成，例如：SQLEx、ActionEx。

### 1.5文件命名规范

外部js文件一律小写命名。

### 1.6命名约定表

|  |  |  |
| --- | --- | --- |
| **操作项** | **命名约定** | **示例** |
| 参数 | 使用传递值/对象的完整的英文描述符。 | userID |
| 字段/属性 | 字段采用完整的英文描述，第一个字母小写，任何中间单词的首字母大写。 | firstName |
| 布尔型的获取成员函数 | 所有的布尔型获取函数必须用单词 is（has）做前缀。 | isString()  hasMoney() |
| 原型 | 采用完整的英文描述符，所有单词的第一个字母大写。 | Customer |
| 异常原型名 | 由表示该异常类型等的单词和Exception组成 | SQLException  ActionException |
| 异常实例名 | 通常采用字母 e 、ex表示异常。  多个异常时使用异常名或其简写加E、Ex等构成 | e  SQLEx |
| 静态常量字段（常量） | 全部采用大写字母，单词之间用下划线分隔。采用静态常量获取成员函数。 | DEFAULT\_NAME |
| 局部变量 | 采用完整的英文描述符，第一个字母小写，但不要隐藏已有字段。例如，如果有一个字段叫 firstName，不要让一个局部变量叫 firstName。 | ,totalMoney |
| 循环计数器 | 通常采用字母 i，j，k 或者 counter，index | i,j,k,count,index |
| 成员函数 | 采用完整的英文描述说明成员函数功能，第一个单词尽可能采用一个生动的动词，除第一个单词外，每个单词第一个字母小写。 | openFile()  addUser() |
| 获取成员函数 | 被访问字段名的前面加上前缀 get。 | getUserName() |
| 设置成员函数 | 被访问字段名的前面加上前缀 set。 | setUserName () |

## 2. 代码组织与风格

代码组织和风格的基本原则是：利于个人开发，便于相互交流。

因个人习惯和编辑器等可以设置而形成自己独特的风格，但必须前后一致，并符合本规范的基本要求和原则。

本章所涉及到的内容一般都可在editplus工具编辑环境中进行相应设置。

### 2.1整体风格

整体风格可以参考使用Vue来实现

### 2.2 缩排

代码中以TAB（2个字符）缩进，在编辑器中请将TAB设置为相同的长度，否则在不同编辑器或设置下会导致TAB长度不等而影响整个程序代码的格式。

所有脚本代码全部左起空2格位开始，如下：

//不要这样顶格写

function test{

…

}

//无论外部.js文件或者页面内嵌脚本，一律左起空2格再写

function test(){

…

}

### 2.3文件布局

外部js文件一般放置在与HTML页面文件平级的js目录下。

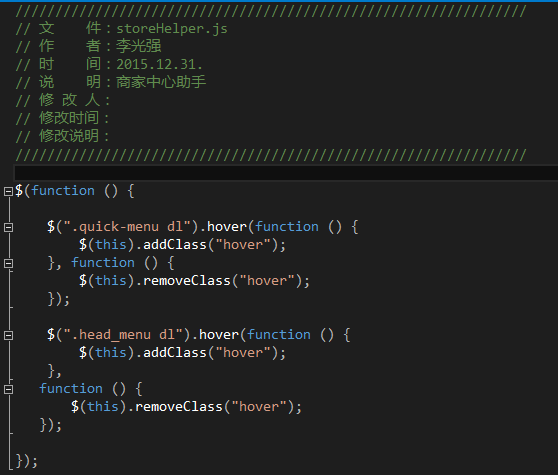
内嵌js，一般应放在HTML标签底部

### 2.4调试行代码处理

如Debugger、console等禁止上传SVN或注释

## 3.注释格式规范

在每个JS文件顶部，必须加入文件说明性注释，示例：



**注意：**其它基本同JAVA篇的注释格式规范章节。

## 4.编程规范

javascript是一种基于对象的、解释型的脚本语言，它本身不如JAVA那样严格，因此编程规范上可以相对简化一些。可以参考java编程规范。

### 4.1原型

每个原型都应该定义toString()方法以便输出有用信息。

toString()方法示例：

/ \*\*

\* 函数说明：输出Tree基本信息

\*/

function Tree(){

…

}

Tree.prototype.toString = function(){

return “[Tree]”;

}

### 4.2方法

#### 基本原则

方法内不定义新的方法，方法功能唯一。方法的修改需有注释说明（日期、修改人、修改概述），避免在一个较长的方法里提供多个出口：

//不要使用这种方式，当处理程序段很长时将很难找到出口点

if(condition){

return A;

}else{

return B;

}

//建议使用如下方式

var result = null;

if(condition){

result = A;

}else{

result = B;

}

return result;

#### 参数

避免过多的参数，一般以5个为限。

如果确实需要输入多个参数，可以考虑将参数类型定义成一个原型或object，以成员去描述各个参数。

考虑变更函数参数的因素，也应考虑如上处理，避免函数发生变更（个数、顺序）时需要在所有使用到的地方进行修改。

### 4.3表达式与语句

#### 基本原则

表达式和语句应清晰、简洁，易于阅读和理解，避免使用晦涩难懂的语句。

使用圆括号明确表达式执行优先级。

#### 控制语句

判断中如有常量，则应将常量置与判断式的右侧。如：

if ( 1 == getCount())...

if ( null == user)...

boolean类型判断语句尽量不要条件比较值true/false

//建议使用

if (isCond)...

if (!isCond)...

//不尽量使用

if(true == isCond)…

if (false == isCond)…

if(true!= isCond)…

if…else if…else语句必须使用{}将每个判断条件后的执行语句括起来。

#### 循环语句

* 循环中必须有终止循环的条件或语句，避免死循环。
* 当多层循环嵌套时，计数器变量注意不要有冲突。
* 循环内部不声明变量、定义方法，只调用方法
* 注意循环条件在执行循环过程中是否会发生变化，如果会则必须把循环条件的值在执行循环前获取而不要在每次循环去执行。
* 考虑运行效率问题也应把循环条件值放在循环执行前获取。

### 4.4错误处理

#### 基本原则

* 通常的法则是系统在正常状态并且用户正常操作下，不应产生任何异常。
* 对可预见的错误不进行捕捉。
* 对不可预见或者难以解决（通常由于浏览器BUG造成）错误进行try{…}catch(e){..}捕捉处理。

#### 可预见错误

对可预见的错误不进行捕捉处理，而是在错误发生前通过条件判断避免发生，如：

//若不对div1是否为null进行检查，则在其为null时会抛出缺少对象错误

document.getElementById(“div1”).style.width = 100;

//预先对对象进行检查

var objDiv1 = document.getElementById(“div1”);

if(objDiv1!=null){

objDiv1.style.width = 100;

}

#### 不可预见错误

对不可预见或者因浏览器、脚本解析器BUG造成的难以解决的错误需要进行捕捉处理，如：

try{

xmlhttp.open(“GET”,url,NOT\_ASYNC);

}catch(e){

alert(e.description);

}

对捕捉到的错误一般情况必须给出反馈处理，例如alert。

### 4.5测试与Bug跟踪

#### 基本原则

* 测试不通过的代码不得提交到SVN库或者发布。
* 对于将来可能会引起错误，或者需求变更可能会带来很大改动量的编程方式应尽量预先避免，不能因为图一时省事而不考虑后果。

#### 跟踪和缺陷处理

* 当系统出现Bug时应由该Bug负责人（代码负责人）尽快修改。
* Bug的处理根据其优先级高低和级别高低先后处理。
* Bug级别和优先级别参见《测试手册》。
* 禁止隐瞒Bug。

## 5. JavaScript 对象

JavaScript 中的所有事物都是对象：字符串、数值、数组、函数...

此外，JavaScript 允许自定义对象。

### 5.1 JavaScript 对象

JavaScript 提供多个内建对象，比如 String、Date、Array 等等。

对象只是带有属性和方法的特殊数据类型。

### 5.2访问对象的属性

属性是与对象相关的值。

访问对象属性的语法是：

objectName.propertyName

### 5.3 自定义对象

**创建 JavaScript 对象**

本项目中使用对象构造器定义对象。

#### 实例

function person(firstname,lastname,age,eyecolor)

{

this.firstname=firstname;

this.lastname=lastname;

this.age=age;

this.eyecolor=eyecolor;

}

**创建 JavaScript 对象实例**

一旦您有了对象构造器，就可以创建新的对象实例，就像这样：

var myFather=new person("Bill","Gates",56,"blue");

var myMother=new person("Steve","Jobs",48,"green");

**把属性添加到 JavaScript 对象**

可以通过为对象赋值，向已有对象添加新属性：

假设 personObj 已存在，可以为其添加这些新属性：firstname、lastname、age 以及 eyecolor：

person.firstname="Bill";

person.lastname="Gates";

person.age=56;

person.eyecolor="blue";

x=person.firstname;

**把方法添加到 JavaScript 对象**

方法只不过是附加在对象上的函数。

在构造器函数内部定义对象的方法：

function person(firstname,lastname,age,eyecolor)

{

this.firstname=firstname;

this.lastname=lastname;

this.age=age;

this.eyecolor=eyecolor;

this.changeName=changeName;

function changeName(name)

{

this.lastname=name;

}

}

changeName() 函数 name 的值赋给 person 的 lastname 属性。

myMother.changeName("Ballmer");

**注意：**

JavaScript 是面向对象的语言，但 JavaScript 不使用类。

在 JavaScript 中，不会创建类，也不会通过类来创建对象。

**注释**：for...in 循环中的代码块将针对每个属性执行一次。

**实例**

循环遍历对象的属性：

var person={fname:"Bill",lname:"Gates",age:56};

for (x in person)

{

txt=txt + person[x];

}

## 6. JS命名空间

命名空间namespace（某些语言中叫package），是一个在静态语言中常见的概念。它可以帮助我们更好地整理代码，并可避免命名冲突。可以创建一个简单对象字面量来打包所有的相关函数和变量。这个简单对象字面量模拟了命名空间的作用。

示例：



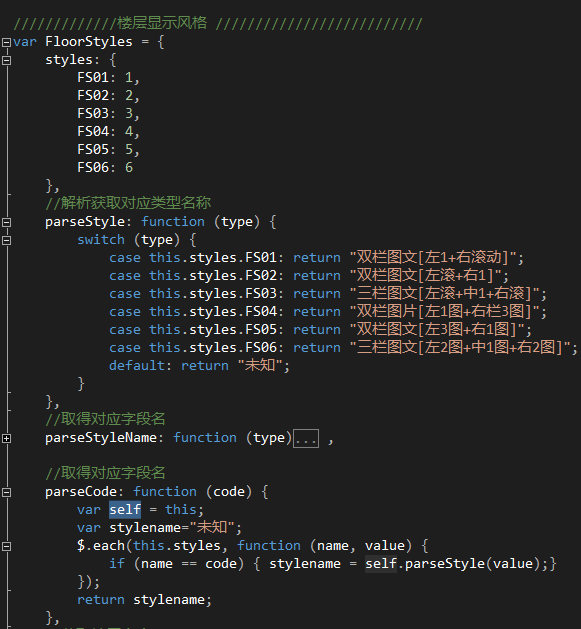
采用命名空间后，代码结构会更加清晰可读。若不想每次都手动构建命名空间。

在项目中，一些常用参数或函数可以使用命名空间加以定义。例如：



**注意：**

在使用命名空间时，要特别注意“this”使用域的变化，在命名空间中再定义函数或调用其它函数时，可能会误用“this”。如：



上示例的parseCode函数中，在$.each操作函数不能使用this操作命名空间中的parseStyle函数，所在parseCode函数定义中使用“self”变量存储“this”，在$.each操作函数中，可以使用self访问“this”中的函数。

# 第四部分SVN控制规范

## 代码上传说明

每次上查代码必须要写提交说明，描述简要提交内容不少于5个字。

## 必要同步时间

上班前先跟新代码，下班前提交代码