HTML/CSS开发规范指南

目录

1. 规范概述
2. 通用约定
   - 分离
   - 文件命名
   - 缩进
   - 编码
   - 小写
   - 注释
   - 待办事项
   - 行尾空格
   - 省略嵌入式资源协议头
   - 代码有效性
3. HTML约定
   - 文档类型
   - 省略type属性
   - 省略属性值
   - 用双引号包裹属性值
   - 嵌套
   - 标签闭合
   - 多媒体替代方案
   - 有效操作
   - 按模块添加注释
   - 格式
   - 语义化标签
   - 模块化
4. 图像约定
   - 图像压缩
   - 背景图
   - 前景图
   - Sprite

规范概述

规范的制定是我们长期以来对工作的积累与沉淀的产物，帮助我们更快、更好、更高效的完成繁重、复杂、多样化的任务，我们制作规范的主要目的在于：

- 降低每个组员介入项目的门槛成本；
- 提高工作效率及协同开发的便捷性；
- 高度统一的代码风格；
- 提供一整套HTML、CSS解决方案，来帮助开发人员快速做出高质量的符合要求的页面。

通用约定

分离

结构（HTML）、表现（CSS）、行为分离（JavaScript）

> 将结构与表现、行为分离，保证它们之间的最小耦合，这对前期开发和后期维护都至关重要。

文件命名

- CSS模块文件，其文件名必须与模块名一致；

假定有这样一个模块：

```
.m-detail { sRules; }
.m-detail-hd { sRules; }
.m-detail-bd { sRules; }
.m-detail-ft { sRules; }
```

> 那么该模块的文件名应该为：`m-detail.css`

- CSS页面文件，其文件名必须与HTML文件名一致；

> 假定有一个HTML页面叫 `product.html`，那么其相对应的CSS页面文件名应该为：`product.css`

> 假定现在有一个 `product.html`，里面有2个模块：

```
+<section class="m-list">
+<section class="m-info">
```

那么开发人员能快速找到与该页面相关的3个直接CSS文件，包括：`product.css`，`m-list.css`，`m-info.css`

缩进

推荐使用tab（2个空格宽度）来进行缩进，可以在IDE里进行设置

编码

- 以 UTF-8 无 BOM 编码作为文件格式；
- 在HTML中文档中用 `<meta charset="utf-8" />` 来指定编码；
- 为每个CSS文档显示的定义编码，在文档首行定义 `@charset "utf-8";`

> 在 Sass 中，如果文档中出现中文，却未显示定义编码，将会编译出错，为了统一各种写法，且提前规避错误几率，统一要求每个CSS文档都需要定义编码

小写

- 所有的HTML标签必须小写；
- 所有的HTML属性必须小写；
- 所有的样式名及规则必须小写。

注释

尽可能的为你的代码写上注释。解释为什么要这样写，它是新鲜的方案还是解决了什么问题。

行尾空格

删除行尾空格，这些空格没有必要存在

省略嵌入式资源协议头（可选）

省略图像、媒体文件、样式表和脚本等URL协议头部声明 (http: , https:)。如果不是这两个声明的URL则不省略。

省略协议声明，使URL成相对地址，防止内容混淆问题和导致小文件重复下载（这个主要是指http和https交杂的场景中）。

不推荐：

```
<script src="http://www.google.com/js/gweb/analytics/autotrack.js"></script>
```

推荐：

```
<script src="//www.google.com/js/gweb/analytics/autotrack.js"></script>
```

不推荐：

```
.example {
  background: url(http://www.google.com/images/example);
}
```

推荐：

```
.example {
  background: url(//www.google.com/images/example);
}
```

> 注：省略协议头在IE7-8下会有一点小问题，外部CSS文件（link和@import）会被下载两遍，所以该条目的约定看具体项目。

代码有效性

- 使用 W3C HTML Validator 来验证你的HTML代码有效性；
- 使用 W3C CSS Validator 来验证你的CSS代码有效性。

> 代码验证不是最终目的，真的目的在于让开发者在经过多次的这种验证过程后，能够深刻理解到怎样的语法或写法是非标准和不推荐的，即使在某些场景下被迫要使用非标准写法，也可以做到心中有数。

HTML约定

文档类型

- 统一使用HTML5的标准文档类型：`<!DOCTYPE html>`；

> HTML5文档类型具备前后兼容的特质，并且易记易书写

- 在文档doctype申明之前，不允许加上任何非空字符；

> 任何出现在doctype申明之前的字符都将使得你的HTML文档进入非标准模式

- 不允许添加 `<meta>` 标签强制改变文档模式。

> 避免出现不可控的问题

省略type属性

在调用CSS和JavaScript时，可以将type属性省略不写

不允许：

```
<link type="text/css" rel="stylesheet" href="base.css" />
<script type="text/javascript" src="base.js"></script>
```

应该：

```
<link rel="stylesheet" href="base.css" />
```

```
<script src="base.js"></script>
```

> 因为HTML5在引入CSS时，默认type值为text/css；在引入JavaScript时，默认type值为text/javascript

省略属性值

非必须属性值可以省略

不允许：

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

应该：

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

> 这里的 readonly 和 disabled 属性的值是非必须的，可以省略不写，我们知道HTML5表单元素新增了很多类似的属性，如: required

用双引号包裹属性值

所有的标签属性值必须要用双引号包裹，同时也不允许有的用双引号，有的用单引号的情况

不允许：

```
<a href=http://www.qunar.com class=home>去哪儿网</a>
```

应该：

```
<a href="http://www.qunar.com" class="home">去哪儿网</a>
```

嵌套

所有元素必须正确嵌套

- 不允许交叉；

不允许：

```
<span><dfn>交叉嵌套</span></dfn>
```

应该：

```
<span><dfn>交叉嵌套</dfn></span>
```

- 不允许非法的子元素嵌套。

不允许：

```
<ul>
        <h3>xx列表</h3>
        <li>asdasdsdasd</li>
        <li>asdasdsdasd</li>
</ul>
```

应该：

```
<div>
        <h3>xx列表</h3>
        <ul>
                <li>asdasdsdasd</li>
                <li>asdasdsdasd</li>
        </ul>
</div>
```

- 不推荐inline元素包含block元素；

不推荐：

```
<span>
        <h1>这是一个块级h1元素</h1>
        <p>这是一个块级p元素</p>
</span>
```

推荐：

```
<div>
        <h1>这是一个块级h1元素</h1>
        <p>这是一个块级p元素</p>
</div>
```

> 规则可参考：

> HTML4/XHTML1.0 Strict: 嵌套规则。
>
> HTML5: 嵌套规则
>
> 举个例子，在HTML5中，a元素同时属于 Flow content, Phrasing content, Interactive content, Palpable content 4个分类，那些子元素是 phrasing 元素的元素可以是 a 的父元素，a 允许的子元素是以它的父元素允许的子元素为准，但不能包含 interactive 元素。

标签闭合

所有标签必须闭合

不允许：

```
<div>balabala...
<link rel="stylesheet" href="*.css">
```

应该：

```
<div>balabala...</div>
<link rel="stylesheet" href="*.css" />
```

> 虽然有些标记没有要求必须关闭，但是为了避免出错的几率，要求必须全部关闭，省去判断某标记是否需要关闭的时间

多媒体替代方案

- 为img元素加上alt属性；
- 为视频内容提供音轨替代；
- 为音频内容提供字母替代等等。

不推荐：

```
<img src="banner.jpg" />
```

推荐：

```
<img src="banner.jpg" alt="520即将到来，爱就大声说出来" />
```

> alt属性的内容为对该图片的简要描述，这对于盲人用户和图像损毁都非常有意义，即无障碍。对于纯粹的装饰性图片，alt属性值可以留空，如 alt=""

有效操作

为表单元素label加上for属性

不允许：

```
<input type="radio" name="color" value="0" /><label>蓝色</label>
<input type="radio" name="color" value="1" /><label>粉色</label>
```

应该：

```
<input type="radio" id="blue" name="color" value="0" /><label for="blue">蓝色</label>
<input type="radio" id="pink" name="color" value="1" /><label for="pink">粉色</label>
```

> for属性能让点击label标签的时候，同时focus到对应的 input 和 textarea上，增加响应区域

按模块添加注释

在每个模块开始和结束的地方添加注释

```
<!-- 新闻列表模块 -->
<div class="m-news g-mod"
...
<!-- /新闻列表模块 -->

<!-- 排行榜模块 -->
<div class="m-topic g-mod"
...
<!-- /排行榜模块 -->
```

> 注释内容左右两边保留和注释符号有1个空格位，在注释内容内不允许再出现中划线"-"，某些浏览器会报错。
>
> 注释风格保持与原生HTML的语法相似：成对出现 `<!-- comment --><!-- /comment -->`

格式

- 将每个块元素、列表元素或表格元素都放在新行；
- inline元素视情况换行，以长度不超过编辑器一屏为宜；
- 每个子元素都需要相对其父级缩进（参见缩进约定）。

不推荐：

```
<div><h1>asdas</h1><p>dff<em>asd</em>asda<span>sds</span>dasdasd</p></div>
```

推荐：

```
<div>
        <h1>asdas</h1>
        <p>dff<em>asd</em>asda<span>sds</span>dasdasd</p>
</div>
```

语义化标签

- 根据HTML元素的本身用途去使用它们；
- 禁止使用被废弃的用于表现的标签，如 center, font 等；
- 部分在XHTML1中被废弃的标签，在HTML5中被重新赋予了新的语义，在选用时请先弄清其语义，如:b, i, u等。

不允许：

```
<p>标题</p>
```

应该：

```
<h1>标题</h1>
```

> 虽然使用p标签，也可以通过CSS去定义它的外观和标题相同，但p标签本身的并不是表示标题，而是表示文本段落

> 参阅：HTML5 Elements

模块化

- 每个模块必须有一个模块名；
- 每个模块的基本组成部分应该一致；
- 模块的子节点类名需带上模块名（防止模块间嵌套时产生不必要的覆盖）；
- 孙辈节点无需再带模块名。

代码如：

```
<section class="m-detail">
        <header class="m-detail-hd">
                <h1 class="title">模块标题</h1>
        </header>
        <div class="m-detail-bd">
                <p class="info">一些实际内容</p>
        </div>
        <footer class="m-detail-ft">
                <a href="#" class="more">更多</a>
        </footer>
</section>
```

> 其中 `.m-detail-hd`，`.m-detail-bd`，`.m-detail-ft` 为可选，视具体模块情况决定是否需要抽象为这种 **头，中，尾** 的结构

图像约定

图像压缩

所有图片必须经过一定的压缩和优化才能发布

背景图

- 使用PNG格式而不是GIF格式，因为PNG格式色彩更丰富，还能提供更好的压缩比；
- 在需要兼容IE6的项目中，尽可能选择PNG8，而不是使用PNG24+滤镜。

前景图

- 内容图片建议使用JPG，可以拥有更好地显示效果；
- 装饰性图片使用PNG。

Sprite

- CSS Sprite是一种将数个图片合成为一张大图的技术（既可以是背景图也可以是前景图），然后通过偏移来进行图像位置选取；
- CSS Sprite可以减少http请求。

结语

坚持一致性的原则。

一个团队的代码风格如果统一了，首先可以培养良好的协同和编码习惯，其次可以减少无谓的思考，再次可以提升代码质量和可维护性。

统一的代码风格，团队内部阅读或编辑代码，将会变得非常轻松，因为所有组员都处在一致思维环境中。