HTML编码规范

1 前言

本文档的目标是使HTML代码风格保持一致，容易被理解和被维护。

2 代码风格

2.1 缩进与换行

[强制] 使用 `4` 个空格做为一个缩进层级，不允许使用 `2` 个空格 或 `tab` 字符。

示例：

```
<ul>
    <li>first</li>
    <li>second</li>
</ul>
```

[建议] 每行不得超过 `120` 个字符。

解释：

过长的代码不容易阅读与维护。但是考虑到 HTML 的特殊性，不做硬性要求。

2.2 命名

[强制] `class` 必须单词全字母小写，单词间以 `-` 分隔。

[强制] `class` 必须代表相应模块或部件的内容或功能，不得以样式信息进行命名。

示例：

```
<!-- good -->
<div class="sidebar"></div>

<!-- bad -->
<div class="left"></div>
```

[强制] 元素 `id` 必须保证页面唯一。

解释：

同一个页面中，不同的元素包含相同的 id，不符合 id 的属性含义。并且使用 document.getElementById 时可能导致难以追查的问题。

[建议] `id` 建议单词全字母小写，单词间以 `-` 分隔。同项目必须保持风格一致。

[建议] `id`、`class` 命名，在避免冲突并描述清楚的前提下尽可能短。

示例：

```
<!-- good -->
<div id="nav"></div>
<!-- bad -->
<div id="navigation"></div>

<!-- good -->
<p class="comment"></p>
<!-- bad -->
<p class="com"></p>

<!-- good -->
<span class="author"></span>
<!-- bad -->
<span class="red"></span>
```

[强制] 禁止为了 `hook 脚本`，创建无样式信息的 `class`。

解释：

不允许 class 只用于让 JavaScript 选择某些元素，class 应该具有明确的语义和样式。否则容易导致 css class 泛滥。

使用 id、属性选择作为 hook 是更好的方式。

[强制] 同一页面，应避免使用相同的 `name` 与 `id`。

解释：

IE 浏览器会混淆元素的 id 和 name 属性， document.getElementById 可能获得不期望的元素。所以在对元素的 id 与 name 属性的命名需要非常小心。

一个比较好的实践是，为 id 和 name 使用不同的命名法。

示例：

```
<input name="foo">
<div id="foo"></div>
<script>
// IE6 将显示 INPUT
alert(document.getElementById('foo').tagName);
</script>
````

2.3 标签

[强制] 标签名必须使用小写字母。

示例：

```html
<!-- good -->
<p>Hello StyleGuide!</p>

<!-- bad -->
<P>Hello StyleGuide!</P>
```

[强制] 对于无需自闭合的标签，不允许自闭合。

解释：

常见无需自闭合标签有input、br、img、hr等。

示例：

```
<!-- good -->
<input type="text" name="title">

<!-- bad -->
<input type="text" name="title" />
```

[强制] 对 `HTML5` 中规定允许省略的闭合标签，不允许省略闭合标签。

解释：

对代码体积要求非常严苛的场景，可以例外。比如：第三方页面使用的投放系统。

示例：

```
<!-- good -->
<ul>
    <li>first</li>
    <li>second</li>
</ul>

<!-- bad -->
<ul>
    <li>first
    <li>second
</ul>
```

[强制] 标签使用必须符合标签嵌套规则。

解释：

比如 div 不得置于 p 中，tbody 必须置于 table 中。

详细的标签嵌套规则参见HTML DTD中的 `Elements` 定义部分。

[建议] `HTML` 标签的使用应该遵循标签的语义。

解释：

下面是常见标签语义

- p - 段落
- h1,h2,h3,h4,h5,h6 - 层级标题
- strong,em - 强调
- ins - 插入
- del - 删除
- abbr - 缩写
- code - 代码标识
- cite - 引述来源作品的标题
- q - 引用
- blockquote - 一段或长篇引用
- ul - 无序列表
- ol - 有序列表
- dl,dt,dd - 定义列表

示例：

```
<!-- good -->
<p>Esprima serves as an important <strong>building block</strong> for some JavaScript language
tools.</p>

<!-- bad -->
<div>Esprima serves as an important <span class="strong">building block</span> for some
JavaScript language tools.</div>
```

[建议] 在 `CSS` 可以实现相同需求的情况下不得使用表格进行布局。

解释：

在兼容性允许的情况下应尽量保持语义正确性。对网格对齐和拉伸性有严格要求的场景允许例外，如多列复杂表单。

[建议] 标签的使用应尽量简洁，减少不必要的标签。

示例：

```
<!-- good -->
<img class="avatar" src="image.png">

<!-- bad -->
<span class="avatar">
    <img src="image.png">
</span>
```

2.4 属性

[强制] 属性名必须使用小写字母。

示例：

```
<!-- good -->
<table cellspacing="0">...</table>

<!-- bad -->
<table cellSpacing="0">...</table>
```

[强制] 属性值必须用双引号包围。

解释：

不允许使用单引号，不允许不使用引号。

示例：

```
<!-- good -->
<script src="esl.js"></script>

<!-- bad -->
<script src='esl.js'></script>
<script src=esl.js></script>
```

[建议] 布尔类型的属性，建议不添加属性值。

示例：

```
<input type="text" disabled>
<input type="checkbox" value="1" checked>
```

[建议] 自定义属性建议以 `xxx-` 为前缀，推荐使用 `data-`。

解释：

使用前缀有助于区分自定义属性和标准定义的属性。

示例：

```
<ol data-ui-type="Select"></ol>
```

3 通用

3.1 DOCTYPE

[强制] 使用 `HTML5` 的 `doctype` 来启用标准模式，建议使用大写的 `DOCTYPE`。

示例：

```
<!DOCTYPE html>
```

[建议] 启用 **IE Edge** 模式。

示例：

```
<meta http-equiv="X-UA-Compatible" content="IE=Edge">
```

[建议] 在 `html` 标签上设置正确的 **lang** 属性。

解释：

有助于提高页面的可访问性，如：让语音合成工具确定其所应该采用的发音，令翻译工具确定其翻译语言等。

示例：

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

3.2 编码

[强制] 页面必须使用精简形式，明确指定字符编码。指定字符编码的 `meta` 必须是 `head` 的第一个直接子元素。

解释：

见 HTML5 Charset能用吗 一文。

示例：

```
<html>
    <head>
        <meta charset="UTF-8">
        ......
    </head>
    <body>
        ......
    </body>
</html>
```

[建议] `HTML` 文件使用无 `BOM` 的 `UTF-8` 编码。

解释：

UTF-8 编码具有更广泛的适应性。BOM 在使用程序或工具处理文件时可能造成不必要的干扰。

3.3 CSS和JavaScript引入

[强制] 引入 `CSS` 时必须指明 `rel="stylesheet"`。

示例：

```
<link rel="stylesheet" src="page.css">
```

[建议] 引入 `CSS` 和 `JavaScript` 时无须指明 `type` 属性。

解释：

`text/css` 和 `text/javascript` 是 type 的默认值。

[建议] 展现定义放置于外部 `CSS` 中，行为定义放置于外部 `JavaScript` 中。

解释：

结构-样式-行为的代码分离，对于提高代码的可阅读性和维护性都有好处。

[建议] 在 `head` 中引入页面需要的所有 `CSS` 资源。

解释：

在页面渲染的过程中，新的CSS可能导致元素的样式重新计算和绘制，页面闪烁。

[建议] `JavaScript` 应当放在页面末尾，或采用异步加载。

解释：

将 script 放在页面中间将阻断页面的渲染。出于性能方面的考虑，如非必要，请遵守此条建议。

示例：

```
<body>
    <!-- a lot of elements -->
    <script src="init-behavior.js"></script>
</body>
```

[建议] 移动环境或只针对现代浏览器设计的 **Web** 应用，如果引用外部资源的 `URL` 协议部分与页面相同，建议省略协议前缀。

解释：

使用 `protocol-relative URL` 引入 CSS，在 `IE7/8` 下，会发两次请求。是否使用 `protocol-relative URL` 应充分考虑页面针对的环境。

示例：

```
<script src="//s1.bdstatic.com/cache/static/jquery-1.10.2.min_f2fb5194.js"></script>
```

4 head

4.1 title

[强制] 页面必须包含 `title` 标签声明标题。

[强制] `title` 必须作为 `head` 的直接子元素，并紧随 `charset` 声明之后。

解释：

title 中如果包含 ascii 之外的字符，浏览器需要知道字符编码类型才能进行解码，否则可能导致乱码。

示例：

```
<head>
    <meta charset="UTF-8">
    <title>页面标题</title>
</head>
```

4.2 favicon

[强制] 保证 `favicon` 可访问。

解释：

在未指定 favicon 时，大多数浏览器会请求 Web Server 根目录下的 favicon.ico 。为了保证favicon可访问，避免404，必须遵循以下两种方法之一：

1. 在 Web Server 根目录放置 favicon.ico 文件。
2. 使用 link 指定 favicon。

示例：

```
<link rel="shortcut icon" href="path/to/favicon.ico">
```

5 图片

[强制] 禁止 `img` 的 `src` 取值为空。延迟加载的图片也要增加默认的 `src`。

解释：

src 取值为空，会导致部分浏览器重新加载一次当前页面，参考：https://developer.yahoo.com/performance/rules.html#emptysrc

[建议] 避免为 `img` 添加不必要的 `title` 属性。

解释：

多余的 title 影响看图体验，并且增加了页面尺寸。

[建议] 为重要图片添加 `alt` 属性。

解释：

可以提高图片加载失败时的用户体验。

[建议] 添加 `width` 和 `height` 属性，以避免页面抖动。

[建议] 有下载需求的图片采用 `img` 标签实现，无下载需求的图片采用 `CSS` 背景图实现。

解释：

1. 产品 logo、用户头像、用户产生的图片等有潜在下载需求的图片，以 img 形式实现，能方便用户下载。
2. 无下载需求的图片，比如：icon、背景、代码使用的图片等，尽可能采用 css 背景图实现。

6 表单

6.1 控件标题

[强制] 有文本标题的控件必须使用 `label` 标签将其与其标题相关联。

解释：

有两种方式：

1. 将控件置于 label 内。
2. label 的 for 属性指向控件的 id。

推荐使用第一种，减少不必要的 id。如果 DOM 结构不允许直接嵌套，则应使用第二种。

示例：

```
<label><input type="checkbox" name="confirm" value="on"> 我已确认上述条款</label>

<label for="username">用户名：</label> <input type="textbox" name="username" id="username">
```

6.2 按钮

[强制] 使用 `button` 元素时必须指明 `type` 属性值。

解释：

button 元素的默认 type 为 submit，如果被置于 form 元素中，点击后将导致表单提交。为显示区分其作用方便理解，必须给出 type 属性。

示例：

```
<button type="submit">提交</button>
<button type="button">取消</button>
```

[建议] 尽量不要使用按钮类元素的 `name` 属性。

解释：

由于浏览器兼容性问题，使用按钮的 name 属性会带来许多难以发现的问题。具体情况可参考此文。

7 模板中的 HTML

[建议] 模板代码的缩进优先保证 `HTML` 代码的缩进规则。

示例：

```
<!-- good -->
{if $display == true}
<div>
    <ul>
    {foreach $item_list as $item}
        <li>{$item.name}<li>
    {/foreach}
    </ul>
</div>
{/if}

<!-- bad -->
{if $display == true}
    <div>
        <ul>
    {foreach $item_list as $item}
        <li>{$item.name}<li>
    {/foreach}
        </ul>
    </div>
{/if}
```

[建议] 模板代码应以保证 `HTML` 单个标签语法的正确性为基本原则。

示例：

```
<!-- good -->
<li class="{if $item.type_id == $current_type}focus{/if}">{ $item.type_name }</li>

<!-- bad -->
<li {if $item.type_id == $current_type} class="focus"{/if}>{ $item.type_name }</li>
```

[建议] 在循环处理模板数据构造表格时，若要求每行输出固定的个数，建议先将数据分组，之后再循环输出。

示例：

```
<!-- good -->
<table>
    {foreach $item_list as $item_group}
    <tr>
        {foreach $item_group as $item}
        <td>{ $item.name }</td>
        {/foreach}
    <tr>
    {/foreach}
</table>

<!-- bad -->
<table>
<tr>
    {foreach $item_list as $item}
    <td>{ $item.name }</td>
        {if $item@iteration is div by 5}
    </tr>
    <tr>
        {/if}
    {/foreach}
</tr>
</table>
```