**前端代码规范 及 最佳实践**

**概述**

本文档包含了Isobar公司的创意技术部（前端工程）开发web应用的规范。现在我们把它开放给任何希望了解我们迭代过程最佳实践的人。

编写本文档的主要驱动力是两方面： 1) 代码一致性 以及 2) 最佳实践。 通过保持代码风格和传统的一致性，我们可以减少遗留系统维护的负担，并降低未来系统崩溃的风险。而通过遵照最佳实践，我们能确保优化的页面加载、性能以及可维护的代码。



**总体指导原则**[◊](http://coderlmn.github.io/code-standards/#_general_guidelines)[**返回顶部**](http://coderlmn.github.io/code-standards/)

**前端开发核心思想**[◊](http://coderlmn.github.io/code-standards/#_pillars_of_front_end_development)

1. 表现、内容和行为的分离
2. [标记应该是结构良好、语义正确](http://www.bbc.co.uk/guidelines/futuremedia/technical/semantic_markup.shtml) 以及 [普遍合法](http://coderlmn.github.io/code-standards/#_appendix_a_validators) 。
3. [Javascript应该起到渐进式增强用户体验的作用](http://icant.co.uk/articles/pragmatic-progressive-enhancement/) 。

**总体原则**[◊](http://coderlmn.github.io/code-standards/#_general_practices)

缩进[◊](http://coderlmn.github.io/code-standards/#_indentation)

对于所有编程语言，我们要求缩进必须是软tab（用空格字符）。在你的文本编辑器里敲 *Tab* 应该等于 **4个空格** 。

可读性 vs 压缩[◊](http://coderlmn.github.io/code-standards/#_readability_vs_compression)

对于维护现有文件，我们认为可读性比节省文件大小更重要。大量空白和适当的ASCII艺术都是受鼓励的。任何开发者都不必故意去压缩HTML或CSS，也不必把Javascript代码最小化得面目全非。

我们会在服务器端或build过程中自动最小化并gzip压缩所有的静态客户端文件，例如CSS和JS。



**标记**[◊](http://coderlmn.github.io/code-standards/#_markup)[**返回顶部**](http://coderlmn.github.io/code-standards/)

任何网页的首要组件就是基于标签的HTML标记语言。超文本标记语言 (HTML) 曾有一段不堪回首的历史，但最近几年已经是**皇 上 回 宫**了。经过对它基于XML的XHTML变种的漫长试验之后，整个行业终于接受了HTML代表web的未来这一事实。

标记定义了文档的结构和纲要，并提供结构化的内容。除了最基本的概念（例如标题、段落和列表）之外，标记并不是用来定义页面上内容的外观和体验的。HTML的表现属性都已经被废弃了，有关样式的定义应该被包含在 [样式表](http://coderlmn.github.io/code-standards/#css) 中。

**HTML5**[◊](http://coderlmn.github.io/code-standards/#_html5)

[HTML5](http://www.ibm.com/developerworks/library/wa-webstandards/index.html) 是HTML 和 XHTML 的新版本。 在 [HTML5 草案](http://www.w3.org/TR/html5/) 的规范中定义了可以用 HTML 和 XML编写的单一的语言，意在解决在之前HTML的迭代中发现的一些问题并满足*web应用*的需求，这是以前HTML没有充分覆盖到的领域（[来源](http://html5.org/) ） 。

在合适的时候，我们会使用HTML5 Doctype 和 HTML5 特性。

我们会对照 [W3C 校验器](http://validator.w3.org/) 测试我们的标记，以确保标记是结构良好的。我们的目标并不是100%的合法代码，但是校验肯定对于编写可维护的站点以及调试代码都大有帮助。 **Isobar公司不保证代码都是100%合法，而是确信跨浏览器体验会相当一致** 。

模板[◊](http://coderlmn.github.io/code-standards/#_template)

对HTML5文件，我们使用 [H5BP](http://html5boilerplate.com/) 针对我们自己项目需求修改的一个分支。 你也可以[从这里Fork H5BP](https://github.com/isobar-idev/html5-boilerplate)。

**Doctype**[◊](http://coderlmn.github.io/code-standards/#_doctype)

标记中必须总是使用合适的Doctype来指示浏览器触发标准模式. 永远要避免Quirks模式。

HTML5特别好的一个方面就是它简化了Doctype需要的代码。无意义的属性被弃用了，DOCTYPE 声明也被显著地简化了。另外，也无需再用 CDATA 对内联JavaScript代码进行转义，而这在此之前为了让XHTML符合XML的严密性是必需的。

HTML5 Doctype

1.<!DOCTYPE html>

字符编码[◊](http://coderlmn.github.io/code-standards/#_character_encoding)

所有标记必须通过UTF-8编码传递，因为这种编码方式是对国际化最友好的。必须在HTTP的header和HTML文档的head部分都指定这种编码。

设定字符编码是通过 <meta> 标签进行。

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

如果是HTML5，则只需要写：

1.<meta charset="utf-8">

**标记的总体原则**[◊](http://coderlmn.github.io/code-standards/#_general_markup_guidelines)

下面是编写HTML标记的总体原则。提醒大家一点，在创建的HTML文档里总是要使用能够代表内容语义的标记。

* 段落分隔符要使用实际对应的<p>元素，而不是用多个<br>标签。
* 在合适的条件下，充分利用<dl> （定义列表）和<blockquote> 标签。
* 列表中的条目必须总是放置于<ul>、<ol>或<dl> 中，永远不要用一组 <div>或<p> 来表示。
* 给每个表单里的字段加上 <label> 标签，其中的 for 属性必须和对应的输入字段对应，这样用户就可以点击标签。同理，给标签加上 cursor:pointer; 样式也是明智的做法。 [讨论 1](http://www.accessifyforum.com/viewtopic.php?t=1926#14178)[讨论 2](http://www.accessifyforum.com/viewtopic.php?t=6738)
* 不用使用输入字段中的 size 属性。该属性是和输入字段里文本的 font-size 相关的。应该使用CSS宽度。
* 在某些闭合的 </div> 标签旁边加上一段html注释，说明这里闭合的是什么元素。这在有大量嵌套和缩进的情况下会很有用。
* 不要把表格用于页面布局。
* 在合适的条件下，利用 [microformats](http://en.wikipedia.org/wiki/Microformat) 和/或者 Microdata ，具体说是 hCard 和 adr。
* 在合适的条件下，利用 <thead>、<tbody>和<th>标签 (以及Scope属性）。

具备恰当语法结构(THEAD,TBODY,TH [scope])的 Table 标记

01.<table summary="This is a chart of year-end returns for 2005.">

02.<thead>

03.<tr>

04.<th scope="col">Table header 1</th>

05.<th scope="col">Table header 2</th>

06.</tr>

07.</thead>

08.<tbody>

09.<tr>

10.<td>Table data 1</td>

11.<td>Table data 2</td>

12.</tr>

13.</tbody>

14.</table>

* 对于页眉和标题，永远使用首字母大写格式。不要在标记中使用全部大写或小写的标题，而是应用CSS属性 text-transform:uppercase/lowercase 。

属性加引号[◊](http://coderlmn.github.io/code-standards/#_quoting_attributes)

在HTML5规范里并没有严格要求属性值两边加引号。但考虑到一些属性可以接受空白值，为了保持一致性，我们要求所有属性值必须加上引号。

1.<p class="line note" data-attribute="106">This is my paragraph of special text.</p>



**CSS**[◊](http://coderlmn.github.io/code-standards/#_css)[**返回顶部**](http://coderlmn.github.io/code-standards/)

网页的第二个组件就是在层叠样式表（CSS）中包含的表现信息。Web浏览器成功实现CSS后，整整一代web开发者对他们网站的外观和体验拥有了全部的控制权。

正如网页信息在语义方面由 [HTML 标记](http://coderlmn.github.io/code-standards/#markup) 描述， CSS 从表现方面则是通过对视觉属性的定义来描述网页。 CSS 的强大之处在于，这些属性可以混合并通过各种标示符匹配，它可以通过样式规则的分层（”层叠“）来控制页面的布局和视觉特征。

编码总体原则[◊](http://coderlmn.github.io/code-standards/#_general_coding_principles)

* 从外部文件加载CSS，尽可能减少文件数。加载标签必须放在文件的 HEAD 部分。
* 用 LINK 标签加载，[永远不要用@import](http://blog.amodernfable.com/2008/01/21/thoughts-on-linking-to-stylesheets/)。加载样式表

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

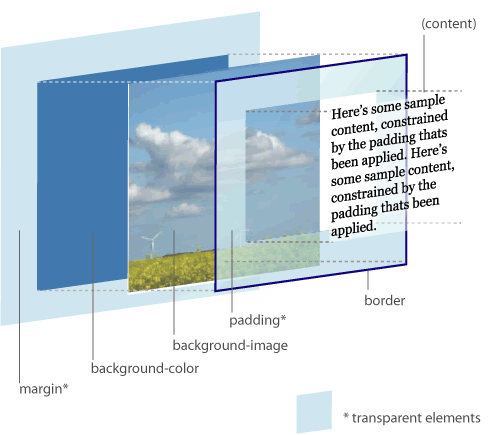
不要用内联式样式

1.<p style="font-size: 12px; color: #FFFFFF">This is poor form, I say</p>

* 不要在文件中用内联式引入的样式，不管它是定义在样式标签里还是直接定义在元素上。这样会很难追踪样式规则。
* 使用 [normalize.css](http://necolas.github.com/normalize.css/) 让渲染效果在不同浏览器中更一致。
* 使用类似 [YUI fonts.css](http://developer.yahoo.com/yui/fonts/) 的字体规格化文件。
* 定义样式的时候，对样式在页面只出现一次的元素用id，其他的用class。
* 理解 [层叠和选择器的明确度](http://www.stuffandnonsense.co.uk/archives/css_specificity_wars.html) ，这样你就可以写出非常简洁且高效的代码。
* 编写性能优化的选择器。尽可能避免使用开销大的CSS选择器。例如，避免 \* 通配符选择器，也不要叠加限定条件到 ID 选择器（例如 div#myid）或 class 选择器（例如 table.results）上。这对于速度至上并包含了成千上万个DOM元素的web应用来说尤为重要。更多相关内容请参阅 MDN 上的这篇 [《编写高效CSS》](https://developer.mozilla.org/en/Writing_Efficient_CSS)。

CSS盒子模型[◊](http://coderlmn.github.io/code-standards/#_css_box_model)

深入学习和理解CSS及基于浏览器的盒子模型，对于掌握CSS布局的基础是非常必要的。

3D CSS 盒子模型示意图， 由 [Jon Hicks](http://hicksdesign.co.uk/boxmodel/) 绘制。

CSS 校验[◊](http://coderlmn.github.io/code-standards/#_css_validation)

我们一般不用 [W3C 校验器](http://jigsaw.w3.org/css-validator/)。

CSS 格式[◊](http://coderlmn.github.io/code-standards/#_css_formatting)

最低要求：选择器单独占一行，每个属性占一行。属性声明要有缩进。

作为提高的要求，关联或孩子样式要增加2-4个空格的缩进。这样有利于分层查看和组织，产生（对某些人来说）可读性更好的样式表。

另外，在给一个样式指定多个选择器的时候，把每个选择器单独放一行是个好主意。这样可以避免一行变得太长，也能提高可读性及版本控制流程。

01..post-list li a{

02.color:#A8A8A8;

03.}

04..post-list li a:hover{

05.color:#000;

06.text-decoration:none;

07.}

08..post-list li .author a,

09..post-list li .author a:hover{

10.color:#F30;

11.text-transform:uppercase;

12.}

在多个开发者协作环境下，避免用单行CSS格式，因为这样会给版本控制带来问题。

字母排序[◊](http://coderlmn.github.io/code-standards/#_alphabetize)

如果你对性能情有独钟， [对CSS属性进行字母排序有利于在GZIP压缩中识别大量可重复的特征](http://www.barryvan.com.au/2009/08/css-minifier-and-alphabetiser/)。

Classes vs. IDs[◊](http://coderlmn.github.io/code-standards/#_classes_vs_ids)

对于所用的样式只出现一次的元素，给它设一个id属性。这个属性只会应用于该元素，不会用到其他地方。Class属性则可以用到多个具有相同样式属性的元素上。具有相同外观和表现的元素可以具有相同的class名。

1.<ul id="categories">

2.<li class="item">Category 1</li>

3.<li class="item">Category 2</li>

4.<li class="item">Category 3</li>

5.</ul>

选择器命名的惯例[◊](http://coderlmn.github.io/code-standards/#_naming_conventions_for_selectors)

无论是 ID 还是 class，对任何东西最好总是根据 **它是什么** 而不是 **它看上去是什么样子** 来命名。 比如一个页面上的特别提示的 class 名是 bigBlueText （大蓝字），可它的样式早就被改成红色小字体，这个名字就没意义了。使用更聪明的惯例如 noteText （提示文字）就好多了，因为即使视觉样式改变了，它也还是管用的。

选择器[◊](http://coderlmn.github.io/code-standards/#_selectors)

[CSS3 选择器](http://www.w3.org/TR/2009/PR-css3-selectors-20091215/) 规格引入了一整套对于更好地选择元素极其有用的 [CSS 选择器](http://www.w3.org/TR/css3-selectors/#selectors)。

伪类[◊](http://coderlmn.github.io/code-standards/#_pseudo_classes)

[伪类](http://www.w3.org/TR/css3-selectors/#pseudo-classes) 使你能动态地修饰网页内容的样式。有些伪类从CSS1 （:visited, :hover等） 和 CSS2 （:first-child, :lang）那时候开始就有了。CSS3又往列表里加入了16个新的伪类，这些伪类对于动态地修饰网页内容的样式特别有用。 [学习如何深度使用伪类](http://www.smashingmagazine.com/2011/03/30/how-to-use-css3-pseudo-classes/)。

组合及属性选择器[◊](http://coderlmn.github.io/code-standards/#_combinators_amp_attribute_selectors)

[组合选择器](http://www.w3.org/TR/css3-selectors/#combinators) 提供了为特定元素选择其后代元素、孩子元素或兄弟元素的快捷方式。

[属性选择器](http://www.w3.org/TR/css3-selectors/#attribute-selectors) 适用于具有特定属性 和/或 特定值的元素。正则表达式的知识对属性选择大有帮助。

明确度[◊](http://coderlmn.github.io/code-standards/#_specificity)

浏览器会通过 [计算选择器的明确度](http://www.w3.org/TR/2009/PR-css3-selectors-20091215/) 来决定应用哪个CSS规则。如果两个选择器都适用于同样的元素，**具有更高明确度的那个会胜出**。

ID 选择器比属性选择器明确度高，class 选择器比任何数量的元素选择器明确度高。尽量使用 ID 选择器来提高明确度。有时候我们可能会想方设法给一个元素应用一条CSS规则，但用尽方法也不能如愿。这种情况有可能是因为我们使用的选择器比另外一个的明确度低，所以明确度高的另一个选择器里的属性就比我们想应用的选择器优先了。这种情况在更大或更复杂的样式表里更为常见。在小一些的项目里，通常这不是大问题。

计算明确度[◊](http://coderlmn.github.io/code-standards/#_calculating_specificity)

当你在一个很大很复杂的样式表上干活的时候，知道如何计算选择器的明确度会有很大帮助，会节约你的时间，并让你的选择器更有效率。

明确度的计算方式是对你的CSS的各种组件计数，并用 (a,b,c,d) 格式表达出来。

* 元素，伪元素: d = 1 – (0,0,0,1)
* 类，伪类，属性: c = 1 – (0,0,1,0)
* Id: b = 1 – (0,1,0,0)
* 内联样式: a = 1 – (1,0,0,0)

不过，也许使用现成的明确度计算器更好一些。

* [明确度计算器](http://specificity.keegan.st/)
* [你应该了解的关于明确度的一些事](http://www.smashingmagazine.com/2007/07/27/css-specificity-things-you-should-know/)
* [IE 明确度 bugs](http://www.brunildo.org/test/IEASpec.html#a)

使用 !important 会覆盖掉所有的明确度，不管它有多高。因此我们倾向于避免使用它。大部分时候是没必要用它的。即使你需要覆盖某个你访问不到的样式表里的选择器，往往也会有其他的办法去覆盖。尽可能避免使用它。

像素 vs. Em[◊](http://coderlmn.github.io/code-standards/#_pixels_vs_ems)

我们使用 px 作为定义 font size 的度量单位，因为它能提供对文本的绝对控制。我们知道为字体大小使用 em 单位一度很流行，这样可以解决 IE6 无法改变基于像素的文本大小的问题。不过，现在所有的主流浏览器（包括 IE7 和 IE8）都支持基于像素单位的文本大小 和/或 整页缩放。既然 IE6 被广泛认为已废弃，用像素定义文本尺寸更好。另外，无单位的 line-height 也应该优先考虑，因为它不会从父元素继承一个百分比值，而是基于 font-size 的一个乘数。

正确

1.#selector {

2.font-size: 13px;

3.line-height: 1.5;  /\*  13 \* 1.5 = 19.5 ~ Rounds to 20px. \*/

4.}

不正确

1./\*  Equivalent to 13px font-size and 20px line-height, but only if the browser default text size is 16px. \*/

2.#selector {

3.font-size: 0.813em;

4.line-height: 1.25em;

5.}

IE Bugs[◊](http://coderlmn.github.io/code-standards/#_internet_explorer_bugs)

不可避免地，当所有其他浏览器看起来都正常工作的时候，各种版本的IE浏览器就会冒出一些莫名其妙的bug，让部署一拖再拖。虽然我们鼓励排除问题，产出无需打补丁就能在所有浏览器上运行的代码，有时候为了在样式表中使用CSS钩子，还是有必要用到CSS if IE 条件注释。[从 paulirish.com 了解更多信息。](http://paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neither/)

修复 IE

1.<!--[if lt IE 7 ]> <body class="ie6"> <![endif]-->

2.<!--[if IE 7 ]>    <body class="ie7"> <![endif]-->

3.<!--[if IE 8 ]>    <body class="ie8"> <![endif]-->

4.<!--[if IE 9 ]>    <body class="ie9"> <![endif]-->

5.<!--[if !IE]><!--> <body> <!--<![endif]-->

1..box { float: left; margin-left: 20px; }

2..ie6 .box { margin-left: 10px; }

如果你在用HTML5（以及 [HTML5 Boilerplate](http://coderlmn.github.io/code-standards/#h5bp)）， 我们推荐使用 [Modernizer](http://www.modernizr.com/) JavaScript库和下列模式：

1.<!--[if lt IE 7]> <html class="no-js ie ie6" lang="en"> <![endif]-->

2.<!--[if IE 7]>    <html class="no-js ie ie7" lang="en"> <![endif]-->

3.<!--[if IE 8]>    <html class="no-js ie8" lang="en"> <![endif]-->

4.<!--[if IE 9]>    <html class="no-js ie9" lang="en"> <![endif]-->

5.<!--[if gt IE 9]><!--><html class="no-js" lang="en"><!--<![endif]-->

速记格式[◊](http://coderlmn.github.io/code-standards/#_shorthand)

一般情况下要优先使用CSS速记格式，一是因为它的简洁，二是用它也可以扩充已有的值，例如margin和padding的情况。 开发者必须注意TRBL 缩写，它表示元素的各边按顺时针方向定义的顺序：上、右、下、左。如果*bottom*没有定义，它就会从*top*继承值。同理，如果*left*未定义，它从*right*继承值。如果只有*top*的值有定义，所有的边都会继承那一个值。

下面是关于减少样式表代码冗余和使用CSS速记格式的更多内容：

* <http://qrayg.com/journal/news/css-background-shorthand>
* <http://sonspring.com/journal/css-redundancy>
* <http://dustindiaz.com/css-shorthand>

图片[◊](http://coderlmn.github.io/code-standards/#_images)

* 对于（用于背景的）重复图片，要使用 [比 1x1 像素大的图片](http://www.iandevlin.com/blog/2010/03/webdev/fading-issue-with-repeating-background-transparent-image-in-internet-explorer)
* 永远不要用空白图片。
* 多使用 [CSS精灵（sprites）](http://coderlmn.github.io/code-standards/#_leverage_css_sprites)。它会使悬停状态更简单，改善页面加载速度，并减少二氧化碳的排放。
* 一般情况下，所有的图片都应该切成带透明背景（PNG8），并裁减成紧密贴合图片外边框。
  + 不过，logo必须总是带有背景遮片，并在裁减内容之外留出内边框。

颜色管理[◊](http://coderlmn.github.io/code-standards/#_color_management)

* 确认团队所有成员都有一致的颜色管理设置。
  + 任意两台显示器显示的颜色很可能会有所不同，但必须使用sRGB颜色作为缺省配置。
  + 当你在Photoshop打开文件时，要注意颜色配置警告，当Photoshop建议把图片转换到另一个配置时，要通知其他团队成员。
* 永远不要把颜色配置嵌入到图片里。
  + 当你从Photoshop保存图片时，务必去掉"Embed Color Profile"选项的勾选。

通用的文本和字体样式[◊](http://coderlmn.github.io/code-standards/#_general_text_and_font_styling)

标题[◊](http://coderlmn.github.io/code-standards/#_headings)

* 要给 h1-h6 标题 -- 包括作为链接的标题 -- 定义缺省样式。在你的CSS文档顶部定义它们，在必要时修改它们以保持整个站点的一致性。
* 标题必须有层次，能表明从大到小不同级别的重要性，h1具有最大的字体大小。
* SEO：要大致地了解页面的层次组织和阅读效果，在开发者工具里关闭CSS效果，你会看到一个基于文字的视图，包括所有的 h1-h6 ， strong， em 等标签。

链接[◊](http://coderlmn.github.io/code-standards/#_links)

* 必须定义链接的缺省样式，样式要和主要的文字样式不同，载悬停状态下也要有不同的样式。
* 当给链接加下划线样式时，使用 border-bottom 并用 text-decoration: none; 加点内边框。这样看起来更好一些。

**Web字体**[◊](http://coderlmn.github.io/code-standards/#_web_typography)

近些年来越来越流行使用web上的定制字体和字型。随着本地浏览器对其支持度的攀升，以及一些支持它的服务和API的出现，这个领域发展的势头很猛。这些方法都各有利弊。项目启动前最好是在技术和版权限制方面先做一些研究，以便为特定项目选择合适的方法。

所有这些方法都有代码开销、开发时间和性能（时间计算和用户感受）的不足。你需要熟悉这些问题，并和团队成员及用户沟通，这样会减少项目后期的大量问题。

下面列出一些内嵌定制字体的流行手段，按我们实施时的优先级排序。

@font-face[◊](http://coderlmn.github.io/code-standards/#_font_face)

[@font-face 规则](http://www.w3.org/TR/2011/WD-css3-fonts-20110324/#font-face-rule) 允许你自定义字体。它最早是在CSS2 规范里定义的，但从CSS2.1被删除了。现在，它是CSS3草案中的推荐稿。

对于web定制字体，我们的首选和最偏爱的选择都是 @font-face ，就是因为它被列入了CSS字体模块工作草案的一部分，这意味着它会随着浏览器支持的提升变得越来越流行，并且随着它不断改善变得更稳定，使用起来也更简单。

就现在而言，使用 @font-face 的时候，建议为每种字体格式定义它的source。这很重要 -- 如果你想让它在大多数浏览器中有效 -- 虽然这不是使用它的必要条件。

在规范中包括的字体格式有：

* **woff**: WOFF (Web Open Font 格式)
* **ttf**: TrueType
* **ttf**, **otf**: OpenType
* **eot**: 嵌入式 OpenType
* **svg**, **svgz**: SVG 字体

防弹 @font-face[◊](http://coderlmn.github.io/code-standards/#_bulletproof_font_face)

为了实现完全的浏览器兼容性，可以使用 Fontsprings 的新版 [防弹 @font-face 语法](http://www.fontspring.com/blog/further-hardening-of-the-bulletproof-syntax) （*2011年2月21日的最新版本*）。

01.@font-face {

02.font-family: 'MyFontFamily';

03.src: url('myfont-webfont.eot');                     /\* IE9 Compat Modes \*/

04.src: url('myfont-webfont.eot?iefix') format('eot'), /\* IE6-IE8 \*/

05.url('myfont-webfont.woff') format('woff'),     /\* Modern Browsers \*/

06.url('myfont-webfont.ttf') format('truetype'),      /\* Safari, Android, iOS \*/

07.url('myfont-webfont.svg#svgFontName') format('svg'); /\* Legacy iOS \*/

08.font-weight: <font-weight>;

09.font-style: <font-style>;

10.// etc.

11.}

这里有一个使用上述技术实现的 [演示](http://www.thecssninja.com/demo/css_fontface/) 。

跨浏览器兼容[◊](http://coderlmn.github.io/code-standards/#_cross_browser_compatibility)

Safari, IE 6-9, IE 9 兼容模式, Firefox, Chrome, iOS, Android, Opera

阻止兼容模式[◊](http://coderlmn.github.io/code-standards/#_prevent_compatibility_mode)

有时候 IE 会在用户不知道的情况下自作主张切换到兼容模式。要阻止你的站点缺省进入兼容模式，可以在站点的<head> 部分加入下列代码：

1.<meta http-equiv="X-UA-Compatible" content="IE=edge">

有关 @font-face 的提示[◊](http://coderlmn.github.io/code-standards/#_tips_for_font_face)

* IE 6–8 只会接受打包成 EOT 的 TrueType 字体。
* font-weight 和 font-style 在 @font-face 里有不同的含义。带有 font-weight:bold; 的声明意味着它是这种字型的粗体版本，而不是直接给文本加粗。
* [@font-face 的一些坑](http://paulirish.com/2010/font-face-gotchas/)

**优点**

* 易于实施
* 多种多样APIs
* 可定制
* 易于加入元素中
* 除了 CSS 之外无需其他库或接口
* 被列入了CSS字体模块3工作草案的一部分

**不足**

* 使用不当的时候，支持它的浏览器有限
* 某些现代浏览器（Chrome, Opera）的老版本并不总是能正常渲染。文本可能会出现毛刺。*\*\*我本人没有亲自证实过，不确定现在这个问题是否还存在。*

Google WebFonts API 和 Font 加载器[◊](http://coderlmn.github.io/code-standards/#_google_webfonts_api_amp_font_loader)

使用 [Google Webfonts](https://code.google.com/apis/webfonts/) 有两套可选方案。这两套方案当然也各有其弊端，但它们也可以用得和@font-face 一样好，这全取决于项目的需要。

Webfonts API[◊](http://coderlmn.github.io/code-standards/#_webfonts_api)

[Google的 Webfonts API](https://code.google.com/apis/webfonts/docs/getting_started.html) 本质上和 @font-face 做的是同样的事情，只是它把所有困难的工作都帮你做好了，提供了更广泛的浏览器支持。这个方案主要的缺点是它使用的字体库非常小。使用它的时候你只需要引入下面这套样式表并指定字体名。

1.<link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Font+Name">

然后给你想应用的选择器定义一个样式即可：

1.CSS selector {

2.font-family: 'Font Name', serif;

3.}

Webfont 加载器[◊](http://coderlmn.github.io/code-standards/#_webfont_loader)

Google 提供的另一个备选方案是 [Webfont 加载器](https://code.google.com/apis/webfonts/docs/webfont_loader.html) ，它是一个 JavaScript 库，提供比字体 API 更多的控制。你也可以像Typekit那样使用多套web字体。 使用这套方案需要把下面的script引入你的页面：

01.<script type="text/javascript">

02.WebFontConfig = { google: { families: [ 'Tangerine', 'Cantarell' ]} };

03.(function() {

04.var wf = document.createElement('script');

05.wf.src = ('https:' == document.location.protocol ? 'https' : 'http') +

06.'://ajax.googleapis.com/ajax/libs/webfont/1/webfont.js';

07.wf.type = 'text/javascript';

08.wf.async = 'true';

09.var s = document.getElementsByTagName('script')[0];

10.s.parentNode.insertBefore(wf, s);

11.})();

12.</script>

用这种方式引入 webfont.js 文件速度更快，如果没有用到前面的Ajax接口的话。否则你应该用下面的方法：

1.<script type="text/javascript" src="https://www.google.com/jsapi"></script>

2.<script type="text/javascript">

3.google.load("webfont", "1");

4.google.setOnLoadCallback(function() {

5.WebFont.load({ google: { families: [ 'Tangerine', 'Cantarell' ]} });

6.});

7.</script>

通过使用这套 Webfont 加载器，你具备更多的定制能力，包括使用更多的字体，而不局限于不大的Google Webfont字库。不过，这种方案需要加载Javascript，这就是为了便利付出一些其他的牺牲了。

**优点**

* 非常易于实施
* 广泛的浏览器支持
* 可与 Typekit 组合
* 如果使用 font 加载器，可以定制
* API 和 @font-face 功能相同

**不足**

* API只提供很小的字体库
* Webfont 加载器需要 JavaScript 才能工作
* 大部分浏览器会先加载页面的其他部分，在这些字体所在部分留下空白，或在有回退选项存在的时候显示回退样式，直到页面加载完成为止。
* webfont 库里的一些字体在 Windows 下渲染效果不佳

Cufon[◊](http://coderlmn.github.io/code-standards/#_cufon)

如果你选择使用 Cufon，我们强烈推荐你使用 [Cufon 压缩版](http://cufon.shoqolate.com/js/cufon-yui.js)。你会需要用 [生成器](http://cufon.shoqolate.com/generate/) 来转换你的字体。

1.<script src="cufon-yui.js" type="text/javascript"></script>

2.<script src="YourFont.font.js" type="text/javascript"></script>

3.<script type="text/javascript">

4.Cufon.replace('h1');        // Works without a selector engine

5.Cufon.replace('#sub1');     // Requires a selector engine for IE 6-7

6.</script>

我们推荐慎重使用 Cufon ，因为如果应用到大量的文本上，它会产生很多开销。访问 [Cufon Wiki](https://github.com/sorccu/cufon/wiki/) 可以获取更多相关信息。

**优点**

* 广泛的浏览器支持
* 在支持它的浏览器中渲染效果好
* 可定制
* 易于实施

**不足**

* 需要 JavaScript 才能工作
* 使用该方法的文本不能被选中
* 并不是适用于所有文字
* 定制可能会很复杂
* 并不总是容易应用到多个元素，特别是在增加类似悬停的效果的时候

Typekit[◊](http://coderlmn.github.io/code-standards/#_typekit)

当选择给网站添加定制字体的时候，使用 [Typekit](https://typekit.com/) 有它不容忽视的优势。它具备很强的平台集成，并且是可扩展和流行的服务。 它可以和 Google Webfonts 一起使用，并且易于加入到以 WordPress, Posterous, Typepad, 和其他类似的 CMS 实现的网站上。

但是，要完整地应用 [Typekit 也不是无本生意](https://typekit.com/plans)。如果你需要把它用到超过2个站点，或用在一个浏览量很高的站点上，你需要每年付49.99美元费用，对于百万以上浏览量的站点费用会加倍。不过，如果你的网站有这么大浏览量，对这点成本你应该是不差钱的。如果不是这样，你可能需要重新考虑你的商业模式了。

**优点**

* 庞大的字体库，包括 Adobe 字体
* 易于实施
* 和Google Webfont API 以及 博客平台 集成
* 免费计划有限制但不会过期

**不足**

* 需要 JavaScript 才能工作
* 免费计划能用到的字体库是有限制的
* 免费和最便宜的计划只允许用到1-2个网站，每个网站只能用2-5种字体。
* 用到超过一个网站就需要付费了

可扩展 Inman Flash 替换 (sIFR)[◊](http://coderlmn.github.io/code-standards/#_scalable_inman_flash_replacement_sifr)

我们不推荐使用这种方法，但是因为它的广泛使用，我们觉得还是有必要介绍它，以便你在选择定制web字体方法时做出胸有成竹的决定。

即使它在web设计师中广为流行，而且它在大多数浏览器中也有很好的支持，使用它的缺点还是大过它的便利性。 弃用 sIFR 的最大也是最明显的原因是它需要使用Flash这一事实。而且，即便是为了让Flash正常工作就需要 JavaScript，而且在你使用此方法渲染的文本能在页面上可见之前，相关的脚本必须都先被加载。更不用说，它会增加页面加载时间，并会让一个慢网站变得更慢。

我们会让你对这些问题算算细账。

**优点**

* 文本可以被选中
* 大多数浏览器都支持
* 在支持它的浏览器上渲染得还不错

**不足**

* 它用到了 Flash
* 需要 JavaScript 来让 Flash 正常工作
* 它是 Flash!
* 脚本加载完之前，文字不会出现
* ...还有，它是 Flash...

字体版权[◊](http://coderlmn.github.io/code-standards/#_font_licensing)

即使你可以把任何字体转换为web字体文件，你还是应该确认这样做是否合法。很多厂商更新了他们关于字体在web上使用的条款。查看 [字体版权和保护的细则](http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=UNESCO_Font_Lic) 获取更多信息。

字体描述及字体文件格式[◊](http://coderlmn.github.io/code-standards/#_specifications_amp_font_file_formats)

* [CSS 2 字体](http://www.w3.org/TR/1998/REC-CSS2-19980512/fonts.html#font-descriptions) – 1998年5月 (已废弃)
* [CSS 3 字体](http://www.w3.org/TR/css3-fonts/) – 2009 工作草案
* [CSS 字体模块](http://www.w3.org/TR/css3-fonts/) – W3C 工作草案 2011年5月
* [WOFF Font Format](http://www.w3.org/TR/WOFF/) – 工作草案 2010
* [SVG 字体格式](http://www.w3.org/TR/SVG/fonts.html)
* [嵌入式 OpenType (EOT) 文件格式](http://www.w3.org/Submission/EOT/)
* [微软 OpenType 规范](http://www.microsoft.com/typography/otspec/)
* [OpenType 特性文件规范](http://www.adobe.com/devnet/opentype/afdko/topic_feature_file_syntax.html#9.e)
* [苹果 TrueType 手册](http://developer.apple.com/fonts/TTRefMan/)

使用 CSS3[◊](http://coderlmn.github.io/code-standards/#_use_css3)

利用CSS3规范中增加的特性，你可以做很多新奇的事情，不过里边很多新特性还没有得到所有主流浏览器的完全支持。不过这也不是说它们现在就不能用。对于这些支持不好的特性，有一些回退的库可用，或者有一些其他补丁，用来填补在浏览器对新特性缺乏支持时出现的空白。

还有一些浏览器特定的属性或前缀也可以用来修饰样式。为跨浏览器支持起见，我们推荐使用[Prefixr.com](http://prefixr.com/) 来确保你加入了所有针对不同浏览器的前缀属性。



**JavaScript**[◊](http://coderlmn.github.io/code-standards/#_javascript)[**返回顶部**](http://coderlmn.github.io/code-standards/)

JavaScript 是网页的第三个主要部件。在网页上适当地应用JavaScript 代码，通过绑定事件和控制整体行为层，能够增强整体的用户和基于浏览器的体验。

随着功能强劲的新浏览器撑起基于浏览器的完整web应用，JavaScript 在近年的流行度爆棚。另外，对Javascript的细致运用为全面操控另外两个部件 -- [HTML 标记](http://coderlmn.github.io/code-standards/#markup)和 [CSS](http://coderlmn.github.io/code-standards/#css) -- 提供了手段。现在，在无需刷新整个页面的情况下，页面结构和页面视觉样式都可以被实时操控。

**JavaScript库**[◊](http://coderlmn.github.io/code-standards/#_javascript_libraries)

我们开发新应用主要会用到 [jQuery](http://api.jquery.com/)，不过我们对原生 JavaScript 和所有现代 javascript 库也具有专业经验。

**编码总体原则**[◊](http://coderlmn.github.io/code-standards/#_general_coding_principles)

* 99%的代码必须封装在外部Javascript文件中。这些文件必须在 BODY 标签的尾部引入，让页面的性能最大化。
* 不要依赖于 user-agent 字符串。进行适当的特性检测. （更多信息参见 [深入 HTML5： 检测](http://diveintohtml5.info/detect.html) 和[jQuery 支持文档](http://api.jquery.com/jQuery.support/)）
* 不要使用 document.write()。
* 所有布尔变量的命名必须用 "is" 开头对正条件的测试

1.isValid = (test.value >= 4 && test.success);

* 给变量和函数的命名要有逻辑意义：例如： popUpWindowForAd 就比 myWindow 好多了。
* 不要人为缩短命名到最小。除了传统的 for 循环中的计数器 i 等简化的情况，变量命名必须长到有明确意义。
* 文档必须遵循 [NaturalDocs](http://www.naturaldocs.org/documenting.html) 结构。
* 常量或配置变量（例如动画持续时间等）必须放在文件的顶部。
* 尽力编写可通用化的函数，让它接受参数并返回值。这样有利于充分的代码重用，而且一旦与引入及外部脚本配合起来，能在脚本需要修改时减少开销。例如，相比硬编码一个带有窗口大小、选项和url的弹出式窗口，不如编写一个接受大小、url和选项作为变量的函数。
* 给代码添加注释！这会有利于减少在调试Javascript函数上花费的时间。
* 不要把时间浪费在用 <!-- --> 给你的内联Javascript加注释上，除非你还在关注 Netscape 4。 :)
* 把你的代码组织成一套 [对象常量/单例](http://kaijaeger.com/articles/the-singleton-design-pattern-in-javascript.html)，按照 [模块化模式](http://www.yuiblog.com/blog/2007/06/12/module-pattern/)，或做成 [带构造器的对象](http://mckoss.com/jscript/object.htm)。
* 最小化全局变量 - 你创建的全局变量越少越好。一般来说，用于你的应用命名空间，1会是个好的数字。在描述任何全局变量的时候要明确指认。

1.window.globalVar = { ... }

留空[◊](http://coderlmn.github.io/code-standards/#_white_space)

总的来说，使用留空应该遵循源远流长的英语阅读惯例。 例如，每个逗号和冒号（以及适用的分号）后面要空一格，但在括号内部的左侧和右侧都不要加空格。另外，大括号应该总是和他们前面的参数出现在同一行。

来看看下面的 JavaScript for循环的例子...

正确

1.for (var i = 0, j = arr.length; i < j; i++) {

2.// Do something.

3.}

不正确

1.for ( var i = 0, j = arr.length; i < j; i++ )

2.{

3.// Do something.

4.}

也不正确

1.for(var i=0,j=arr.length;i<j;i++){

2.// Do something.

3.}

plugins.js 和 script.js[◊](http://coderlmn.github.io/code-standards/#_pluginsjs_and_scriptjs)

从 H5BP 开始我们看到了两个文件， plugins.js 和 script.js。本节概述这两个文件的基本用法。

plugins.js[◊](http://coderlmn.github.io/code-standards/#_pluginsjs)

Plugins.js 的用处是存放网站的所有插件代码。相比链接到很多不同的文件，我们可以把插件代码统一放到这个文件里，从而改善性能。这种用法会有也应该有例外。例如，一个超级大的插件，又只是用在一个很少被访问到的页面上，放在单独的一个下载链接里就会更好，这样只会在目标页面被打开的时候才会被访问。不过，大部分情况下，直接把所有插件的最小化版本粘贴到这里以便访问是靠谱的。

下面就是一个样例文件，包括一个小目录。它可以作为所用到插件的随身指南，包括文档URL，使用它们的思路，诸如此类。

01./\* PLUGIN DIRECTORY

02.本文件中出现的插件 [按出场顺序排序]

03.

04.1.) Animate Background Position - http://plugins.jquery.com/project/backgroundPosition-Effect

05.2.) jQuery Easing Plugin - http://gsgd.co.uk/sandbox/jquery/easing/

06.3.) jQuery Ajax Form plugin - http://jquery.malsup.com/form/#download

07.4.) jQuery validation plugin (form validation) - http://docs.jquery.com/Plugins/Validation

08.-password strength

09.5.) Styled Selects (lightweight) - http://code.google.com/p/lnet/wiki/jQueryStyledSelectOverview

10.\*/

11.

12./\*\*

13.\* 1.) Animate Background Position - http://plugins.jquery.com/project/backgroundPosition-Effect

14.\* @author Alexander Farkas

15.\* v. 1.21

16.\*/

17.(function($) {

18.if(!document.defaultView || !document.defaultView.getComputedStyle){ // IE6-IE8

19.//SNIPPED

20.};

21.})(jQuery);

22.

23.

24./\*\*

25.\* 2.) jQuery Easing Plugin (we're not using jQuery UI as of yet) - http://gsgd.co.uk/sandbox/jquery/easing/

26.\*/

27.

28.// t: current time, b: begInnIng value, c: change In value, d: duration

29.jQuery.easing['jswing'] = jQuery.easing['swing'];

30.

31.jQuery.extend( jQuery.easing,

32.{

33.//SNIPPED

34.

35.});

36.;(function($) {

37.$.fn.ajaxSubmit = function(options) {

38.//SNIPPED

39.}

40.})(jQuery);

41.

42./\*

43.\* jQuery Styled Select Boxes

44.\* version: 1.1 (2009/03/24)

45.\* @requires jQuery v1.2.6 or later

46.\*

47.\* Examples and documentation at: http://code.google.com/p/lnet/wiki/jQueryStyledSelectOverview

48.\*

49.\* Copyright (c) 2008 Lasar Liepins, liepins.org, liepins@gmail.com

50.\*

51.\* Permission is hereby granted, free of charge, to any person obtaining a copy

52.\* of this software and associated documentation files (the "Software"), to deal

53.\* in the Software without restriction, including without limitation the rights

54.\* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

55.\* copies of the Software, and to permit persons to whom the Software is

56.\* furnished to do so, subject to the following conditions:

57.\*

58.\* The above copyright notice and this permission notice shall be included in

59.\* all copies or substantial portions of the Software.

60.\*

61.\* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

62.\* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

63.\* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

64.\* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

65.\* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

66.\* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN

67.\* THE SOFTWARE.

68.\*

69.\*/

70.

71.jQuery.fn.styledSelect = function(settings) {

72.//SNIPPED

73.return this;

74.};

Script.js[◊](http://coderlmn.github.io/code-standards/#_scriptjs)

Script.js 的用处是存放网站或应用的代码代码。再次说明，这种方式并不总是最佳解决方案，因为更大的团队 和/或 规模更大、功能更多的项目可以得益于将应用代码分解为模块或功能特性相关的文件。不过，对于较小较简单的应用，以及最初的原型开发，把所有工作集中到 scripts.js 还是靠谱的。

下面是一个简化了的例子，用到了 [基于标记的低调全面的DOM-ready执行](http://paulirish.com/2009/markup-based-unobtrusive-comprehensive-dom-ready-execution/) 模式（【译者注】这里的‘低调’原文是Unobtrusive ，是一种将Javascript从HTML结构抽离的设计概念，避免在HTML标签中夹杂一堆onchange、onclick……等属性去挂载Javascript事件，让HTML与Javascript分离，依MVC的原则将功能权责清楚区分，使HTML也变得结构化容易阅读。），看起来会类似于这样：

01./\* Name: Demo

02.Author: Demo King \*/

03./\*demo namespace\*/

04.demo = {

05.common : {

06.init     : function(){

07.//initialize

08.},

09.finalize : function(){

10.//finalize

11.},

12.config : {

13.prop : "my value",

14.constant : "42"

15.}

16.},

17.mapping : {

18.init     : function(){

19.//create a map

20.},

21.geolocate    : function(){

22.//geolocation is cool

23.},

24.geocode : function(){

25.//look up an address or landmark

26.},

27.drawPolylines : function(){

28.//draw some lines on a map

29.},

30.placeMarker : function(){

31.//place markers on the map

32.}

33.}

34.}

**变量，ID 及 class**[◊](http://coderlmn.github.io/code-standards/#_variables_id_amp_class)

所有的 JavaScript 变量必须写成全小写或驼峰法。一个例外是构造器函数，按惯例是首字母大写。所有CSS里的 id 和 class 声明都必须只用小写。不允许用连接符或下划线。

事件代理[◊](http://coderlmn.github.io/code-standards/#_event_delegation)

在分配低调（unobtrusive）的事件监听器时，通常可接受的做法是把事件监听器直接分派给那些会触发某个结果动作的元素。不过，偶尔也会有多个元素同时符合触发条件，给每个元素都分配事件监听器可能会对性能有负面影响。这种情况下，你就应该改用事件代理了。

从性能角度考虑，jQuery的 [delegate()](http://api.jquery.com/delegate) 优于 [live()](http://api.jquery.com/live)。

调试[◊](http://coderlmn.github.io/code-standards/#_debugging)

即使采用了最好的校验器，浏览器的怪异性也会不可避免地产生一些问题。有几个堪称无价之宝的工具可以帮助优化代码的健全性和加载速度。很重要的一点是，你手头准备好了这些工具，不管你主要用来开发的浏览器是哪个。我们推荐先在Firefox和Safari上做开发，然后用Google Chrome和Opera，最后针对IE用条件判断做一些额外的微调。下面列出的是一些有帮助的调试器和速度分析器：

* **Firefox**: [Firebug](http://getfirebug.com/), [Page Speed](http://code.google.com/speed/page-speed/), [YSlow](http://developer.yahoo.com/yslow/)
* **Safari**: [Web Inspector](http://developer.apple.com/safari/library/documentation/AppleApplications/Conceptual/Safari_Developer_Guide/UsingtheWebInspector/UsingtheWebInspector.html)
* **Google Chrome**: [Developer Tools](http://google.com/chrome/intl/en/webmasters-faq.html#tools)
* **Opera**: [Dragonfly](http://opera.com/dragonfly/)
* **Internet Explorer 6-7**: [Developer Toolbar](http://microsoft.com/downloads/details.aspx?familyid=E59C3964-672D-4511-BB3E-2D5E1DB91038)
* **Internet Explorer 8-10**: [Developer Tools](http://msdn.microsoft.com/en-us/library/dd565625(v=VS.85).aspx)

优化 JavaScript 的特征[◊](http://coderlmn.github.io/code-standards/#_patterns_for_better_javascript)

* 编写可维护的代码
* 单变量模式
* Hoisting：把所有变量声明统一放到函数的起始位置 （在后部声明的变量也会被JS视为在头部定义，由此会产生问题）
* 不要扩充内置原型（虽然给Object(), Function()之类的内置原型增加属性和方法很巧妙，但是会破坏可维护性）
* 不要用隐含的类型转换
* 不要用 eval()
* 用 parseInt() 进行数字转换
* （规范）左大括号的位置
* 构造器首字母大写
* 写注释
* 不要用 void
* 不要用 with 语句
* 不要用 continue 语句
* 尽量不要用位运算

Stoyan Stefanov 的博文包含了以上原则 [并有详细说明](http://net.tutsplus.com/tutorials/javascript-ajax/the-essentials-of-writing-high-quality-javascript/)（【译者注】这篇博文值得一看）。



**易用性**[◊](http://coderlmn.github.io/code-standards/#_accessibility)[**返回顶部**](http://coderlmn.github.io/code-standards/)

[美国政府相关法规508节： 局域网和互联网信息及应用标准](http://www.section508.gov/index.cfm?FuseAction=Content&ID=12#Web)。

— 本公司开发的接口必须符合相关法规508节的要求。

[W3C 的易用性检查点清单](http://www.w3.org/TR/WCAG10/full-checklist.html)。

— 本公司开发的接口必须符合第一优先级指南。

— [WCAG 1.0 指南](http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/#Guidelines)。



**性能**[◊](http://coderlmn.github.io/code-standards/#_performance)[**返回顶部**](http://coderlmn.github.io/code-standards/)

当我们持续把web的能力发挥到极致的时候，让网页在最小开销或等待时间下可用依然是同样重要的问题。下面的章节说明如何优化网页使用户满意。

优化CSS 和 Javascript传输[◊](http://coderlmn.github.io/code-standards/#_optimize_delivery_of_css_and_javascript)

在生产环境传输CSS和Javascript，必须采用很多优化措施：

* 遵循 [Yahoo 性能指南](http://developer.yahoo.com/performance/)
* 使用 [smush.it](http://smush.it/) 对图片进行无损压缩优化。 还有，用 [YSlow](http://developer.yahoo.com/yslow/) 可以把你所有的图片自动压缩（【译者注】YSlow不仅仅提供性能优化工具，还详细说明了性能优化的34条规则，例如不要用CSS表达式、不要用空白的src或href、使用CDN、缓存AJAX响应、不要用过滤器、减少DOM元素数量等等）。
* 不要在HTML中写内联脚本 <script> 块。 它们会阻塞页面渲染操作，对页面加载时间带来破坏性的影响。
* 适当地设置缓存标题。
* 针对静态资源，考虑单独配置一个无cookie的子域。
* CSS 必须放在文档的 <head> 部分, Javascript 必须正好放在 </body> 标签的前面。
* CSS 和 JavaScript 都必须以最小化方式加载。大部分人会使用 [YUI 压缩器](http://developer.yahoo.com/yui/compressor/) 做这件事。
* CSS 和 JavaScript 都必须 [以gzip形式传输](http://code.google.com/speed/page-speed/docs/payload.html#GzipCompression)。
* CSS 必须串接在一起。显然，只有具备相同媒体类型（例如屏幕或打印）的文件才能合在一起。这里的总体目标是在加载页面的时候减少因为依赖关系而产生的HTTP连接数。
* JavaScript 必须串接在一起。虽然用一个AJAX脚本依赖性管理器（类似于 YUI 3 Loader）可能会比较理想，但实施起来还是挺复杂的。 在这里我还是想推荐单次下载站点用到的所有脚本。（当然了，适当的缓存也是必需的，这样可以让文件在合理的时间内尽可能保持在本地）。
* 串接和最小化通常发生在自动build的过程中，这时候要把代码打包用于发布或生产。有很多人用一些工具做这件事，例如 [Ant](http://ant.apache.org/) 或 [Maven](http://maven.apache.org/) 。

优化 JavaScript 执行[◊](http://coderlmn.github.io/code-standards/#_optimize_javascript_execution)

在页面加载的时候，通常会有很多脚本等待执行，但你可以设定优先级。下面的顺序就是基于用户反馈设定的优先级：

1. 改变页面视觉习性的脚本绝对要首先执行。这包括任何的字体调整、盒子布局、或IE6 pngfix。
2. 页面内容初始化
3. 页面标题、顶部导航和页脚的初始化
4. 绑定事件处理器
5. 网页流量分析、Doubleclick，以及其他第三方脚本

借力 CSS 精灵[◊](http://coderlmn.github.io/code-standards/#_leverage_css_sprites)

CSS 精灵（Sprites） 基本上就是把一批图片资源合并成单个图片文件。然后每个部分用 CSSbackground-position 来展现。典型的 CSS 看起来是这样的

1.a.expandbox { display:block; width: 75px; height: 15px; text-indent: -999px; overflow:hidden;

2.background: url(../img/sprite.png) no-repeat -50px -5px;  }

接力 sprites 实现 :hover 悬停效果是很普遍的。这种情况下你会看到类似于这样的定义：

1.a.expandbox:hover { background-position: -50px -20px; }

使用 sprites 可以减少页面大小，也减少了HTTP连接数，这会加速页面加载。 [在 css-tricks.com 上有更多总体性的技术和概述](http://css-tricks.com/css-sprites/)。尽可能地利用CSS 精灵总体来说是一项最佳实践。

除了主要的sprite之外，很多开发者还会使用一个垂直方向的sprite。这个垂直方向的sprite的宽度（以及高度）会**小于或等于100px**，包含了通常一个挨着一个的图标，诸如列表元素的着重号或对应功能的链接和按钮。Yahoo 就用到了一些，[例如这个](http://l.yimg.com/a/i/ww/sp/pa-icons3.gif)。

有个注意事项就是别把sprite弄得太大，不管是高还是宽超过1000px都会导致用掉大量内存。你可以学习一下 [使用sprite的时机及内存占用](http://blog.vlad1.com/2009/06/22/to-sprite-or-not-to-sprite/)，如果需要了解更多关于创建sprite的总体性提示和技巧，请参阅 [Mozilla 开发博客](http://blog.mozilla.com/webdev/2009/03/27/css-spriting-tips/)。

图片格式[◊](http://coderlmn.github.io/code-standards/#_image_formats)

你应该会用到四种主要的图片格式，细节如下：

1. **JPEG.** - 这种格式涵盖了所有基于摄影的图片，例如主页和目录页推销广告图片，或任何颜色数很高的图片。
2. **PNG24.** - 这种格式在Photoshop里用起来很方便，它输出高颜色数图片，并完全支持像素级的渐变透明度。相对来说，它是相当重量级的格式，达到KB级大小。它也是IE6唯一需要执行pngfix的格式。在那种情况下，本公司推荐使用 [DD\_belatedPNG 脚本](http://www.dillerdesign.com/experiment/DD_belatedPNG/) （这是一个 pngfix ，它能修正 PNG24 在 IE6 里冒出灰色或浅蓝色背景的问题。它们并不总是和 background-position 兼容）。

在很多情况下，你可以使用GIF替代PNG24作为对IE6的应变方案。这在需要用PNG24做sprite的情况下尤其适用。所有的 pngfixes 都会特别慢而且开销巨大，所以最好不要用它们。

1. **PNG8.** - 在256色中可以抓取到如此多样的颜色，所以用JPG之前尝试一下PNG是值得的。而且，PNG比GIF有更高的可压缩度（使用类似 pngcrush 和 pngquant的工具）。这种格式支持在几乎所有浏览器中实现渐变透明度，但在IE6中那些半透明像素会显示为100%透明。不过在很多情况下这也够用了。它也无需运行pngfix脚本，所以对于速度是优化的。

Photoshop 无法正确输出这些半透明文件，但是Fireworks 可以。更多细节请参阅：<http://www.sitepoint.com/png8-the-clear-winner/>

1. **透明 GIF 89a.** - GIF 89a 提供透明度的灵活性和广泛的浏览器支持，但也有限制，它不支持渐变透明度和超过256的色深。就我们的经验而言，64的色深就可以提供质量很好的缩略图，并保持相对小的文件大小。

所有低颜色数图片，例如图表或主题图形应该用PNG8格式制作，因为它是这四种格式中具有最高文件大小效率的。PNG8是我们对于大部分网站图形的主要推荐方案。

关于PNG格式、浏览器支持以及各种格式优缺点的详细信息可以在 [这篇优秀的文章](http://calendar.perfplanet.com/2010/png-that-works/) 中找到。

如需进一步优化所有这些格式，从 [Yahoo的 Smush.It](http://smush.it/) 查看图片可以发现缩小它们的办法。

缓存[◊](http://coderlmn.github.io/code-standards/#_caching)

对于静态内容，浏览器应该把文件在本地缓存，在合理的前提下，保留尽可能长的时间。应该较长远期缓存的内容包括：

* CSS 和 JavaScript
* 产品图片
* 主题图形
* favicon.ico
* flash .swf's
* 优惠信息图片（可能缓存时间会略短）

为了最佳缓存效果，利用http头部的Expires。下面是一个远期的Expires头，它告诉浏览器这个响应在2015年5月15日之前都无需更新：

Expires: Thu, 15 Apr 2015 20:00:00 GMT

如果你的服务器是Apache，可以使用 ExpiresDefault 指令设置相对当前日期的失效日期。下面的例子设置失效日期为请求时间的一年之后：

ExpiresDefault "access plus 1 year"

http头部的 Expires 必须设为据现在一个月到一年（远期）之间的值。缓存只适用于那个指定的URL，所以文件名或任何资源的改变都会产生一个新的拷贝。很多开发者使用build过程来给它们的资源增加一个版本号或时间戳。每个随后的build会开始一个全新的缓存版本，让你在使用远期缓存日期时无需担心。 [Google 的这篇文章里有更多关于浏览器缓存的细节信息](http://code.google.com/speed/page-speed/docs/caching.html#LeverageBrowserCaching)。

把资源文件跨域分片[◊](http://coderlmn.github.io/code-standards/#_shard_resources_across_domains)

静态内容当然应该由不同于HTML所在服务器的另一个域提供访问。这是优化的方案，这样 [对所有静态内容的请求就无需额外的cookies头](http://developer.yahoo.com/performance/rules.html#cookie_free)。而且为整个域编写缓存规则也就容易得多了。（同时，当前域的任何子域会继承当前域的cookies，所以使用全新的域是值得的）。

不过，对于足够多的资源（特别是图片），请求数的增加足以让页面加载变慢。很多浏览器对于他们从每个域能并发下载的资源数有比较低的限制。（在IE6和IE7，这个数字仅仅是2）。在这种情况下，我们可以把资源存放在多个子域，例如：

* static1.otherdomain.com
* static2.otherdomain.com
* static3.otherdomain.com

[Google Speed 上更多有关域分片的信息](http://code.google.com/speed/page-speed/docs/rtt.html#ParallelizeDownloads)

避免用IFRAME[◊](http://coderlmn.github.io/code-standards/#_avoid_iframes)

Iframe 是能添加到指定页面的各种元素中上开销最大的一个。它们会阻塞页面让它无法触发onload事件，直到它们加载完成。有时候它们被其他机构用来处理追踪脚本。例如 Doubleclick floodlight 标签就是一个 iframe，管理员可以从他们的管理面板向里面增加追踪像素。在加入iFrame的任何情况下，它应该在window的onload事件被触发之后再动态加入到DOM中。 [更多细节请参阅 Yahoo 的性能站点。](http://developer.yahoo.com/performance/rules.html#iframes)

第三方集成[◊](http://coderlmn.github.io/code-standards/#_3rd_party_integration)

Omniture[◊](http://coderlmn.github.io/code-standards/#_omniture)

我们推荐在页面加载完成（DOM ready 事件或 window 的 load事件）之后，再用 Javascript 把 Omniture Javascript 代码加入DOM中。通过使用这个技术，可以避免外部域脚本的依赖性，这种依赖性会减慢（并可能挂起）页面加载过程。

Flash[◊](http://coderlmn.github.io/code-standards/#_flash)

在所有情况下， Flash 的位置必须有备选的HTML内容，以使SEO值最大化。对于 XML 驱动的 Flash，备选 HTML 内容必须利用同一个 XML 文件，以确保数据的一致性。

所有替换内容必须使用 SWFObject ，但不应加入内联脚本标签。 SWFObject 的初始化必须在 DOM Ready 事件后触发。最小的播放器版本必须设置为最小 v9，以确保 AS3 兼容性。

**跨浏览器性能策略**[◊](http://coderlmn.github.io/code-standards/#_cross_browser_performance_strategy)

谈到浏览器体验，有两个主要的事实：

1. 每个人都想要可能实现的响应性最好的体验。
2. 加到页面上的所有内容都会使它变慢。

那么，基于这两条人生真谛，我们需要通过什么样的步骤让大家满意呢？

与客户一起确定成功的性能指标[◊](http://coderlmn.github.io/code-standards/#_create_success_metrics_with_the_client)

这些指标必须针对你的客户和项目来定制，在线框图布局阶段之前完成。这些目标从技术角度必须是合理的，也是可测试的。

**可能适用的一些目标**

1. 在支持的浏览器中，最慢的那个也必须在5秒钟之内完成从空白缓存到完全加载并初始化的过程。
2. 悬停状态（以及其他实时变化）必须在100毫秒内响应。
3. 动画必须流畅显示，其中跳变发生的时间 < 15%（包括所有浏览器）。

对于加载时间的目标，定义基准系统是很重要的。类似于 [PageTest](http://www.webpagetest.org/) 的工具是个好的选择。另外，目标也可以分开多个页面来定义，例如：访问量最大的两个目标网页（比如主页和支持页面）。

如果客户对于意向设计有比合理目标更激进的目标，就必须在整个项目决策委员会中统一期望值，并让项目组了解性能指标是最关键的。

在设计阶段沟通性能的影响[◊](http://coderlmn.github.io/code-standards/#_communicating_the_performance_impact_during_design_phase)

**内部**

在 IA、IxD 和视觉设计阶段，前端工程师是负责沟通性能对于交互特性的影响或在目标浏览器上采用特定的视觉技术的角色。要给出设计师的限制条件：“如果我们使用Cufon，每个页面上用到定制字体的元素就不能超过10个。”

**外部**

需要设定期望值： **并不是所有的浏览器都有相同的体验**。它们的表现不会彼此相同，指望它们的特性完全一样也不现实。在IE7的体验中放弃一些新的特性也许是合理的。会考虑被放弃的一些特性可能是： *阴影、过渡、圆角、透明度、渐变色。*

**当沟通某件事的影响时**

* 尽可能详细地明确影响程度：“这对页面加载有害” vs “这会在IE中增加2秒的页面加载时间“
* 如有可能，提供快捷的概念验证（Proof of Concept）：”没有siFR的相似页面在2秒钟内加载，而用了siFR的用了8秒，而且在滚动时有延迟“

遵循最佳实践进行开发[◊](http://coderlmn.github.io/code-standards/#_develop_according_to_best_practices)

选择那些性能优化的库和插件。基于性能目标做出明智的架构决定。同时在可能的前提下尽量减少 DOM 操作，设计样式要让页面在加载和初始化的时候 [避免视觉变化](http://paulirish.com/2009/avoiding-the-fouc-v3/) 。

在QA流程中评估性能[◊](http://coderlmn.github.io/code-standards/#_measure_performance_during_qa)

QA 团队也应该把性能相关的因素和视觉、功能和可用性问题放在一起来确定优先级。开发者和 QA 必须确定如何分配优先级。在 QA 过程中，成功的指标必须定期测试。

**测试用的工具**

* [YSlow](http://developer.yahoo.com/yslow/), [Page Speed](http://code.google.com/speed/page-speed/), [Hammerhead](http://stevesouders.com/hammerhead/), [MSFast](http://msfast.myspace.com/), [PageTest](http://www.webpagetest.org/)

**如果性能指标没有达到，有三个选择：**

1. *代码返工* - 重做架构，发现瓶颈，重构代码，优化指标让系统在浏览器里更快执行
2. *放弃该特性* - 只对较慢的浏览器关闭它
3. *重新设计用户交互方式* - 也许新的设计会有一招搞定问题的办法

我们认为，通过这个方法，在应对性能挑战的过程中，项目相关各方都有更好的机会统一期望，共同前进，并形成更加行之有效的工作流程。



**浏览器测试和支持**[◊](http://coderlmn.github.io/code-standards/#_browser_testing_and_support)[**返回顶部**](http://coderlmn.github.io/code-standards/)

今天的用户可以从相当大的范围中选择web浏览器，每种浏览器都提供了略微（或相当）不同的体验。作为开发者，我们的责任恰恰是选择我们创建的网页展示给用户的方式。本节描述我们是如何做出这当中的一些决定的。

我们支持的浏览器[◊](http://coderlmn.github.io/code-standards/#_what_we_support)



**Isobar 支持任何**[**Yahoo 浏览器支持分级**](http://developer.yahoo.com/yui/articles/gbs/)**中列出的A级浏览器，除了Opera之外。对此也可能会有其他的例外，基于地区市场和它们特定的指标**。

我们会努力支持任何客户指定的其他任务关键浏览器 (IE 5.5, Opera, Konqueror, Safari 3 on PC, 等等)，虽然我们不能保证所有功能都可能实现。

我们的测试方式[◊](http://coderlmn.github.io/code-standards/#_how_we_test)

全面的浏览器测试对于每个web项目都是必须的。必须付出大量精力进行跨浏览器和平台测试，以确保质量和一致的用户体验。配置测试环境会是一项挑战，却是值得去做的。

在Microsoft Windows上的测试[◊](http://coderlmn.github.io/code-standards/#_on_microsoft_windows)

IE 测试[◊](http://coderlmn.github.io/code-standards/#_ie_testing)

由于不可能在一台PC上安装多于一个IE浏览器，IE的测试是个挑战。幸好微软最终提供了老版本IE的开发版下载。这些运行拆解版Microsoft Windows的虚拟磁盘时不时地失效（过期）。通常隔几个月就需要重新设置它们。从你的MSD版权（如果有）获取的Microsoft Windows开发版也会是一个选择，取决于你能够获取到的东西。

* [虚拟 PC](http://www.microsoft.com/windows/virtual-pc/default.aspx) - 虚拟PC必须安装在你的计算机上，如果你用的是Windows 7，你必须使用"XP 模式”。
* [Microsoft Windows VPC 映像](http://www.microsoft.com/downloads/en/details.aspx?FamilyID=21eabb90-958f-4b64-b5f1-73d0a413c8ef&displaylang=en) - 有多种虚拟磁盘映像，你可能需要安装多个以构成全面的测试环境，这取决于你的项目。

此外，其他不是那么有效的IE测试选项（通常是不推荐的）包括了 [IETester](http://www.my-debugbar.com/wiki/IETester/HomePage)，它还是好于 [Multiple\_IE](http://tredosoft.com/Multiple_IE)和 [IE7 standalone](http://tredosoft.com/IE7_standalone)。

Firefox[◊](http://coderlmn.github.io/code-standards/#_firefox)

* [Firefox 3.6+](http://www.mozilla.com/en-US/firefox/all-older.html) 也必须安装在本地 - 以及通过 [移动应用](http://portableapps.com/) 获取的 3.0 版本。
* 如果你能胜任， [安装 Firefox 3, 3.5, 和/或 3.6 版，和 FF4 安装在一起](https://wiki.molecular.com/wiki/Install_Firefox_3)。Firefox 配置管理器允许你安装到不同的目录并 [为每个版本保持不同的配置文件](http://lifehacker.com/5481213/master-multiple-firefox-profiles-for-more-productive-browsing)。

PC 版 Safari[◊](http://coderlmn.github.io/code-standards/#_safari_for_the_pc)

* 使用 [最新的PC版Safari版本](http://www.apple.com/safari/). 它和 OSX 版的 Safari 的一致性达到 98% ，但不是完全一致，所以如果它是必需的平台就需要测试。

Opera[◊](http://coderlmn.github.io/code-standards/#_opera)

* 你可以下载 [存档的旧版本](http://arc.opera.com/pub/opera/win/)。要运行多个版本，可以把它们安装到不同的文件夹中。

Google Chrome 和 Chrome 版本[◊](http://coderlmn.github.io/code-standards/#_google_chrome_and_chrome_versions)

Google Chrome 会自动更新，正常情况下绝大部分用户都会有最新版本。要是每种浏览器都这样多好啊。对于Google Chrome就不需要担心旧版本的问题了。

在 Mac OS X上的测试[◊](http://coderlmn.github.io/code-standards/#_on_mac_os_x)

对于核心的Mac OSX 浏览器，Mozilla Firefox, Google Chrome和Apple Safari 提供的浏览体验基本和它们的Windows版本一样。尽管如此，某些操作系统级的差异还是会出现，网站必须在两个平台上进行测试。典型的差异是关于字体渲染，所以有时候会冒出间距问题。

测试在Mac上安装的Windows环境[◊](http://coderlmn.github.io/code-standards/#_testing_for_windows_on_a_mac)

在 Mac OSX 上测试基于Windows的浏览器有几种选择。首先，Mac 提供了一个 "[boot camp](http://en.wikipedia.org/wiki/Boot_Camp_(software))" 分区，它允许你在Mac 上启动一个Microsoft Windows分区。这是一个复杂但完整的测试环境。一旦你用Windows启动，就可以用通常 [Windows 环境下的测试方法](http://coderlmn.github.io/code-standards/#vhds)。

其他选择包括在 Mac OSX 内部虚拟化 Windows，让你可以在Mac OS内部运行Windows。

[Microsoft 虚拟磁盘](http://coderlmn.github.io/code-standards/#vhds) 在这里的大部分选择中是可以打开或转换的，这样就能在一定程度上利用Windows 用户可以用到的那些测试方法。即使你也可以同时在Mac上测试，有些人还是会认为这样更加灵活...

* [Parallels](http://www.parallels.com/products/desktop/) - Parallels 可用，而且可能已经被本公司IT部安装到你的Mac上了。
* [VMWare Fusion](http://www.vmware.com/products/fusion/overview.html) - VMWare Fusion 通过它们的 Fusion 产品提供了同一级别的 Windows 虚拟化。

Mozilla Firefox[◊](http://coderlmn.github.io/code-standards/#_mozilla_firefox)

正如在Windows 上一样，你可以在一台Mac 上安装和运行 Mozilla Firefox 的多个拷贝，虽然通过配置管理器设置多个配置更为复杂一些。尽管如此，你可以通过一些小技巧，[通过Automator程序创建分开的配置并顺利运行它们](http://sonnygill.net/mac/mac-multiple-firefox-profiles/)。

IE standalone 版会出现的bug[◊](http://coderlmn.github.io/code-standards/#_bugs_in_standalone_ie_versions)

注意：IE6 standalone 版在某些情况下对透明度的实现是有bug的。这会导致任何应用于CSS 过滤器的透明度（例如alpha透明度或者24位PNG）失效。在这种情况下必须测试透明度，你会需要本地安装的IE6。

还有人发现安装在Vista 平台上的IE7 和Windows XP 上的IE7 确实有差异，所以，你可能会希望确保团队中的某个人也有这种配置。[IETester](http://www.my-debugbar.com/wiki/IETester/HomePage) 修复了一批这样的问题，和Xenocode 浏览器的做法类似。

**浏览器分辨率**[◊](http://coderlmn.github.io/code-standards/#_browser_resolution)

除了适应各种浏览器，开发者还必须持续注意用户的屏幕分辨率。随着显示器屏幕越来越大，分辨率的广度也随之增加。下面是关于分辨率的一些工作准则。

**1024px 分辨率**

* 估算的折叠位置在 570px 处。
* 优化宽度： 960px - 在两侧留出合适的内边距，960可以被很多数字整除，而且能够很好地配合[IAB 广告的标准宽度](http://www.iab.net/iab_products_and_industry_services/1421/1443/1452)
* 增大宽度： 970px - 在大部分浏览器中还会留出一些内边距。 这个数字和 [黄金比例](http://en.wikipedia.org/wiki/Golden_ratio) 吻合得比较好
* 最大宽度： 996px - 在主流浏览器中还不会产生水平滚动条。 [基于此处的研究](http://www.nealgrosskopf.com/tech/thread.php?pid=43) ，如果你不希望出现水平滚动条，宽度的最大值是 1003px。

**关于窗口大小的当前统计**

* [真没劲 - 1024px 分辨率下的优化宽度？](http://www.cameronmoll.com/archives/001220.html)
* [浏览器、操作系统和搜索引擎的市场占有率](http://marketshare.hitslink.com/report.aspx?qprid=17&qpmr=100&qpdt=1&qpct=3&qptimeframe=M)
* [全球Web 统计](http://www.w3counter.com/globalstats.php)

**不过，系统分辨率和浏览器尺寸并不是一样的**

* [浏览器尺寸很重要 - 实际数据 | mentalized.net](http://mentalized.net/journal/2006/10/24/browser_size_does_matter_actual_numbers/)
* [我需要支持什么样的尺寸 | baekdal.com](http://www.baekdal.com/reports/actual-browser-sizes/abs-sizes/)
* [调查结果： 只有50.4% 的受访者会把浏览器窗口最大化](http://www.456bereastreet.com/archive/200704/poll_results_504_of_respondents_maximise_windows/)
* [屏幕分辨率和页面布局](http://www.useit.com/alertbox/screen_resolution.html)



**搜索引擎优化（SEO）**[◊](http://coderlmn.github.io/code-standards/#_search_engine_optimization)[**返回顶部**](http://coderlmn.github.io/code-standards/)

良好的web设计和开发的一个重要部分就是SEO 。要想确保一个网页不仅能让搜索引擎合适地索引到，而且能让那些只有有限的web能力的人访问到，结构良好的代码是其中的关键。

注意SEO最佳实践[◊](http://coderlmn.github.io/code-standards/#_be_aware_of_seo_best_practices)

* [打印媒体 CSS 的最佳实践](http://24ways.org/2007/back-to-the-future-of-print)。
* 站点/应用 符合 [浏览器分辨率指南](http://coderlmn.github.io/code-standards/#_browser_resolution) 。
* 站点/应用 兼容 [浏览器测试与支持](http://coderlmn.github.io/code-standards/#_browser_testing_and_support) 描述的浏览器需求。
* 注意可用性最佳实践，例如 508法规 和 WCAG 标准
  + [http://www.section508.gov](http://www.section508.gov/)
  + <http://www.w3.org/TR/WCAG20/>.

易于索引[◊](http://coderlmn.github.io/code-standards/#_indexability)

你必须使用语义化的标记，在关闭 Javascript 和 CSS 之后它仍然是可读和逻辑的。所有页面内容必须是HTML 形式；我们不希望使用iframe 或 Javascript 来加载初始的可索引内容。

所有链接都必须指向 HTML

1.<a href="/shelf/jordan/page/2">

而不是这种

1.<a href="javascript:loadPage(2);">

这样既能被搜索引擎正确索引，也能让用户在新窗口或新标签中打开链接。

优化[◊](http://coderlmn.github.io/code-standards/#_optimization)

每个页面的 title 标签应该突出目标关键字。每个页面的 title 应该是独特的。标题（h1,h2,等等）应该构成文档的轮廓并代表该页面最重要的关键字。URL 应该是人类可读的，其中包含主要的目标关键字：

http://domain.com/mens-shoes/basketball/jordan/jordan-mens-ajf-6-basketball-shoe/

**vs**

http:// domain.com/ecomm.cfm?view=prod&prodId=23425

Flash和图片替换[◊](http://coderlmn.github.io/code-standards/#_flash_and_image_replacement)

总是为 flash 使用备选HTML 内容。所有广告促销图片应该使用基于 CSS 的替代图片而不仅仅设定 alt 属性。

>对不支持flash 版本的应变

1.<a href="/nike/morethanagame/" id="morethan">

2.<h4>Nike: More Than A Game</h4>

3.<h5>Experience the movement and view apparel</h5>

4.</a>

1.a#more than { background:url(/promos/nikegame.jpg) no-repeat; width: 200px; height: 100px;

2.text-indent: -999px; overflow:hidden; display:block; }

Google 的 SEO报告卡[◊](http://coderlmn.github.io/code-standards/#_googles_seo_report_card)

[Google 的 SEO报告卡](http://googlewebmastercentral.blogspot.com/2010/03/googles-seo-report-card.html) 是给 Google 的产品团队提供关于通过简单易接受的优化改进他们的产品页面的思路的一项措施。这些优化手段不仅仅是为了帮助搜索引擎更好地理解他们页面的内容，也是改进他们的用户体验。

访问他们的站点时，诸如修复404错误和无效链接、简化URL 选项、提供易于理解的标题以及页面摘录等简单的步骤，其实对用户和搜索引擎都是有利的。



**代码审查**[◊](http://coderlmn.github.io/code-standards/#_code_reviews)[**返回顶部**](http://coderlmn.github.io/code-standards/)

代码审查是确保系统质量和用户体验的正式流程中的基石。这涉及到召开一个由标记编写者、审查者和其他相关人员参加的会议，在会上提交有关材料，产生后续的代码修改要求。简单地说，我们鼓励进行代码审查，磨刀不误砍柴工嘛。

**为啥俺要参加代码审查涅？**

代码审查是用于降低项目风险的战略性时间投资。

经常地，接口开发者被要求根据线框图或视觉构图编写标记。不过，有可能设计的屏幕不能轻易转换为标记，或者转换后质量有损失。代码审查为在页面投入生产之前发现并解决这些风险提供了一个机会。

**代码审查能够提升跨项目的整体知识水平**

既然代码审查涉及到项目内外的成员，这有利于在整个团队中分享技术和最佳实践。

**代码审查能在bug从一些模板繁衍到多个页面之前就封杀它们**

理想情况下，代码审查是在开发过程的早期进行的，早于页面开始全面投入生产。当模板被团队审查并在多个校验工具和浏览器运行，潜在的bug就会冒出来。这是修复bug的理想时机。

**代码审查给不熟悉项目的外部成员提供了发现代码中问题的机会**

项目外部的审查者比在代码上工作了更长时间的标记编写者更容易发现问题。

**哪些人应该参加代码审查？**

归根到底，项目的前端工程主管要负责确保代码审查遵循合适的流程。

理想情况下，一位部门主管应该作为代码审查的主持人，除非部门主管自己正好是被审查代码的接口开发者。在这种情况下，由一位项目经理进行主持。

审查小组应该包括至少两位来自接口技术团队精通开发和最佳实践的资深成员。

**代码审查中有哪些要求？**

在进行代码审查之前，需要审查的模板必须整体完成开发、经过校验、并针对项目需要用到的浏览器和平台进行了测试。

部门主管 和/或 接口开发者必须在代码审查前至少48小时 分发以下材料：

* 所有页面代码，相关的服务器端引用，CSS 和Javascript。这些必须有完整注释，左侧列出行号，在每个打印页的页脚标明文件/页面名称。
* 每个模板的截屏
* 如果适用，标明模板对应的URL
* 项目支持的浏览器和平台的清单
* 已知问题和关注领域的清单

很典型的情况是，直到代码审查进行之前，代码还在不断地修改。不幸的是，这样就没有足够的时间来校验和测试了。如果这种情况发生了，最好是重新安排代码审查的时间以确保其效果。

另外，部门主管 和/或 接口开发者应该预定一间会议室和电话会议号码并提供给所有参与者，因为有可能某些项目组或审查组成员不在现场。用一个小时来审查两三个模板应该足够了；不过，需要的时间也随模板的大小和复杂度而不同。

在代码审查过程中，一位部门主管 和/或 接口开发者应该主持会议，而部门主管或项目经理做记录并分配行动事项。审查者应该在事前审阅过代码并准备好提问或提供反馈意见。

记录和行动事项（包括负责人）应该在代码审查后分发给所有参与者。如果代码审查产生了本质性的变更，或没有完成对所有代码的审查，就有必要安排第二次代码审查。不过，这必须在项目组内讨论以确定其可行性。



**附录**[◊](http://coderlmn.github.io/code-standards/#_appendices)[**返回顶部**](http://coderlmn.github.io/code-standards/)

下面是引用、工具和其他与本文档相关的内容。

（【译者注】下面的内容以链接为主，是一些辅助性的材料，这次就不翻译了。如有不理解的内容，可与本人联系。）

**附录 A: 校验器**[◊](http://coderlmn.github.io/code-standards/#_appendix_a_validators)

* [W3C CSS Validation Service](http://jigsaw.w3.org/css-validator/)
* [HTML Validation firefox extension](https://addons.mozilla.org/en-US/firefox/addon/249)
* [CSS validator](http://jigsaw.w3.org/css-validator/)
* Accessibility - [Bobby Validation Service](http://bobby.watchfire.com/)
  + Tests individual pages for accessibility against either the W3C or Section 508 standards.
* Accessibility - [Cynthia Says Portal](http://www.cynthiasays.com/)
  + Similar to Bobby, tests individual pages for accessibility against either the W3C or Section 508 standards.

**附录 B: 工具**[◊](http://coderlmn.github.io/code-standards/#_appendix_b_tools)

* [Browsershots](http://browsershots.org/) makes screenshots of your web design in different operating systems and browsers
* [Browserling](https://browserling.com/) is an interactive cross-browser testing site. Fully interactive sessions, not static screenshots.
* Accessibility - [Evaluation, Repair, and Transformation Tools for Web Content Accessibility](http://www.w3.org/WAI/ER/existingtools.html) - lists validation tools for validating, testing and fixing web content to make it more accessible.

代码编辑器[◊](http://coderlmn.github.io/code-standards/#_code_editors)

A great code editor can spark productivity in exceptional ways. Many developers prefer rudimentary text editors, others prefer powerful integraded development environments (IDEs). What follows is a general listing of some of the more well-known tools, it would be impossible to list them all.

**Aptana**

Aptana Studio is a powerful, free and open source Ajax development environment which runs stand-alone or within Eclipse. Aptana Studio offers tooling for Ajax including HTML, CSS, DOM, and JavaScript editing and debugging, plus support via additional free plugins for PHP, Ruby on Rails, Adobe AIR, Apple iPhone development. It also features full SVN repository integration for committing, branching, tagging, merging and repository browsing. [Aptana](http://aptana.com/). [Linux, Mac, Windows]

**Geany**

Geany is a text editor using the GTK2 toolkit with basic features of an integrated development environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages. It supports many filetypes and has some nice features. [Geany](http://www.geany.org/). [Linux, Mac, Windows]

**Notepad ++**

Notepad++ is a free (free as in "free speech", but also as in "free beer") source code editor and Notepad replacement, which supports several programming languages, running under the MS Windows environment. [Notepad ++](http://notepad-plus.sourceforge.net/uk/about.php). [Windows]

**e TextEditor**

E is a new text editor for Windows, with powerful editing features and quite a few unique abilities. It makes manipulating text fast and easy, and lets you focus on your writing by automating all the manual work. You can extend it in any language, and by supporting TextMate bundles, it allows you to tap into a huge and active community. [e TextEditor](http://www.e-texteditor.com/). [Windows]

**Edit Plus**

EditPlus is a text editor, HTML editor and programmers editor for Windows. While it can serve as a good Notepad replacement, it also offers many powerful features for Web page authors and programmers. [EditPlus](http://www.editplus.com/). [Windows]

**Homesite**

HomeSite 5.5 provides a lean, code-only editor for web development. Advanced coding features enable you to instantly create and modify HTML, CFML, JSP, and XHTML tags, while enhanced productivity tools allow you to validate, reuse, navigate, and format code more easily. Configure Adobe (formerly Macromedia) HomeSite to fit your needs by extending its functionality and customizing the interface. [Homesite](http://www.adobe.com/products/homesite/). [Windows]

**TextMate**

TextMate claims to be the "Missing Editor" for Mac OS X. A general purpose editor with a sparse interface, the real power is in it's extensibility. Features column selections, recordable macros, snippets, auto-pairing of brackets and other characters, clipboard history, code folding, tab-triggers, tabbed placeholders and mirror typing. And that's just for starters. Anything that can be done via scripts through the Mac command line can be done through custom commands, allowing an extremely high degree of customization and expansion of the feature set. TextMate's "bundle" format has been adapted by many other code editors including the aforementioned e TextEditor. [TextMate](http://www.macromates.com/). [Mac]

**Espresso**

Espresso was created by the same fellow that created the innovative CSSEdit CSS editor. Espresso features syntax highlighting, code folding, code completion, document outliner/navigators, projects, powerful find features, and built-in file transfer publishing capabilities. Finally, it has a powerful "Sugar" feature set which allows the creation of custom commands and plugins. [Espresso](http://macrabbit.com/espresso/). [Mac]

**BBEdit**

BBEdit is the grand-daddy of Mac code editors for web development. Features syntax highlighting, exceptionally powerful text manipulation tools, multi-file searches, a scriptable API, text clippings, and extensive Mac Automator features. [BBEdit](http://www.barebones.com/products/bbedit/). [Mac]

**TextWrangler**

The free "little brother" of BBEdit, it is a powerful raw text editor with a massive text manipulation feature set. Searches, regular expressions, text transformations, syntax highlighting and code navigation tools for a variety of different language environments.[TextWrangler](http://www.barebones.com/products/textwrangler/). [Mac]

**Coda**

Coda from Panic software is a powerful IDE with code editing, terminal, remote file management, and help documentation all built into one UI. Aiming to be a one stop shop for your web development workflow, it also features SVN integration and a new plug-in builder with powerful scripting support and TextMate bundle importing. Finally, code clips and live multi-user editing are also supported. [Coda](http://www.panic.com/coda/). [Mac]

**UltraEdit**

Another editor that's been around for ages, this is an immensely robust and powerful text editor, able to open files limited only by the amount of memory on your computer. The feature list is virtually too much to list, with a massive list of text manipulation features, project support, powerful search and replace, hex editing, function lists, a massive list of languages supported (600+) remote file ftp, telnet, ssh, file comparison, scriptable macros, tools and compiler support, and much, much more. [UltraEdit](http://www.ultraedit.com/products/ultraedit/). [Linux, Mac, Windows]

**Sublime Text**

A relatively new editor, Sublime Text is a new approach in editors. "Open Anything" searches through file names and file contents, with remarkable efficiency. Incredibly powerful selection controls allow editing text in multiple locations at once and the "Minimap" gives you a bird's eye view of the open file so you can find your place easily. Actively being developed, new features are being added and the community around the editor is rapidly expanding. Macros, auto-complete, snippets, build tools, the list of features goes on and on. Supports Linux and Mac starting with version 2. [Sublime Text](http://www.sublimetext.com/). [Linux, Mac, Windows]

**Vim**

If you have to ask, it's probably not for you. [Vim](http://www.vim.org/) [Linux, Mac, Windows]

Google Chrome 扩展[◊](http://coderlmn.github.io/code-standards/#_google_chrome_extensions)

**Developer Tools**

Not actually an extension for Chrome, but built right in (shares much with Safari's Web Inspector, both being derived from WebKit.) This suite of tools features a DOM inspector, basic JavaScript debugger, profiling tools, network loading inspector and timelines, page resources inspectors, and more. [Developer Tools](http://code.google.com/chrome/devtools/docs/overview.html).

**code cola**

A pop-up panel with CSS editing tools for examining and modifying the styles on a given page. [code cola](https://chrome.google.com/webstore/detail/lomkpheldlbkkfiifcbfifipaofnmnkn).

**Firebug Lite for Google Chrome**

You really don't need to install an extension to use Firebug Lite with Chrome, though the extension is nice because it enables one-click application of the Firebug Lite script to any page you are working with. Not the full Firebug feature set, but close. [Firebug Lite](https://chrome.google.com/webstore/detail/bmagokdooijbeehmkpknfglimnifench).

**HTML5 Outliner**

The HTML5 Outliner adds a pop-up with a generated HTML5 outline of the current page's header hierarchy. Helps for checking your pages' organization against the new HTML5 header outlining algorithms. [HTML5 Outliner](https://chrome.google.com/webstore/detail/afoibpobokebhgfnknfndkgemglggomo).

**Pendule**

Nice set of tools for showing data about and interacting with the current web page. Very similar to the [original web developer toolbar for Firefox](http://chrispederick.com/work/web-developer/). [Pendule](https://chrome.google.com/webstore/detail/gbkffbkamcejhkcaocmkdeiiccpmjfdi).

**Web Developer for Chrome**

Chris Pederick, the original developer of the original Web Developer toolbar for Firefox has ported the majority of it over to Chrome. There you have it. [Web Developer](http://chrispederick.com/work/web-developer/).

Firefox 插件[◊](http://coderlmn.github.io/code-standards/#_firefox_plugins)

**FireFTP**

FireFTP is a free, secure, cross-platform FTP client for Mozilla Firefox which provides easy and intuitive access to FTP servers. [FireFTP](http://fireftp.mozdev.org/).

**Firebug**

Firebug integrates with Firefox to put a wealth of development tools at your fingertips while you browse. You can edit, debug, and monitor CSS, HTML, and JavaScript live in any web page. [Firebug](https://addons.mozilla.org/en-US/firefox/addon/1843).

**Firequery**

FireQuery is a collection of Firebug enhancements for jQuery integrated into Firebug.[Firequery](https://addons.mozilla.org/en-us/firefox/addon/firequery/).

**Firecookie**

Firecookie adds cookie viewing, editing, and deletion to Firebug. [Firecookie](https://addons.mozilla.org/en-US/firefox/addon/firecookie/).

**CSS Usage**

CSS Coverage is an extension for Firebug which allows you to scan multiple pages of your site to see which CSS rules are actually used in your site. [CSS Usage](https://addons.mozilla.org/en-us/firefox/addon/css-usage/).

**Greasemonkey**

Allows you to customize the way a web page displays using small bits of JavaScript.[GreaseMonkey](https://addons.mozilla.org/en-US/firefox/addon/748).

**Web Developer Toolbar**

Adds a menu and a toolbar with various web developer tools. [Web Developer Toolbar](https://addons.mozilla.org/en-US/firefox/addon/60).

**JSView**

Adds an item in the status bar that displays all external JavaScript and CSS files loaded on a given page. Allows you to click on and view the files and things like their URLs. Great way to pull file URLs to put into Charles for remote debugging. [JSView](https://addons.mozilla.org/en-US/firefox/addon/2076).

**Live HTTP headers**

When running, captures all HTTP traffic from the browser, which enables you to see what files are being requested as well as information about the requests and server responses.[Live HTTP Headers](https://addons.mozilla.org/en-US/firefox/addon/3829).

**Quick Locale Switcher**

A tremendous help when doing internationalization, Quick Locale Switcher allows you to change the locale sent along in the browser's user-agent HTTP header, telling servers to display content for you in other locales. [Quick Locale Switcher](https://addons.mozilla.org/en-US/firefox/addon/1333).

**Screengrab**

Screengrab sits in the Firefox status bar, allowing you to capture and copy or save screen shots of everything from selections of a web page to the entire page, even parts displayed "below the fold." [Screengrab](https://addons.mozilla.org/en-US/firefox/addon/1146).

**Total Validator**

Enables one-click access to sending your page through a markup validator. No better way to quickly check for missing or mismatched tags! Also available as a standalone application.[Total Validator](http://www.totalvalidator.com/tool/extension.html).

Opera 扩展[◊](http://coderlmn.github.io/code-standards/#_opera_extensions)

[Dragonfly](http://www.opera.com/dragonfly/) is Opera's developer tool similar to Firebug.

IE 插件[◊](http://coderlmn.github.io/code-standards/#_ie_plugins)

**CompanionJS, DebugBar, IE8 Dev tools**.

Charles 代理[◊](http://coderlmn.github.io/code-standards/#_charles_proxy)

[Charles](http://www.charlesproxy.com/) watches all requests and can tell you a lot of information about them. Also supremely useful is **Map Local** which lets you use a local file in place of a given URL (good for replacing a minified js with a full one).

Fiddler[◊](http://coderlmn.github.io/code-standards/#_fiddler)

From the site: ["Fiddler](http://www.fiddler2.com/fiddler2/)is a Web Debugging Proxy which logs all HTTP(S) traffic between your computer and the Internet. Fiddler allows you to inspect traffic, set breakpoints, and "fiddle" with incoming or outgoing data. Fiddler includes a powerful event-based scripting subsystem, and can be extended using any .NET language."

Speedlimit[◊](http://coderlmn.github.io/code-standards/#_speedlimit)

[Speedlimit](http://mschrag.github.com/) is a Leopard (works in snow leopard) preference pane for limiting your network bandwidth to one of a couple different speeds—768k DSL, Edge, 3G, and Dialup. Good for testing your lowest supported speeds or when you want to know how your app will function in real world speeds.

教程和工具[◊](http://coderlmn.github.io/code-standards/#_tutorials_amp_tools)

* [CSS Cheatsheet](http://lesliefranke.com/files/reference/csscheatsheet.html)
* [CSS Links](http://www.dezwozhere.com/links.html)
* [Clean CSS](http://www.cleancss.com/)
* [Fluid Grids Code Generator](http://csswizardry.com/fluid-grids/)

图标[◊](http://coderlmn.github.io/code-standards/#_icons)

* [Famfamfam silk icons](http://www.famfamfam.com/lab/icons/silk/)
* [Sweetie](http://sublink.ca/icons/)
* [Paul's bookmarks for more icons](http://delicious.com/paul.irish/icons)
* [Fugue Icons - 3,100 icons in PNG format.](http://p.yusukekamiyamane.com/)

**附录 C: 资源**[◊](http://coderlmn.github.io/code-standards/#_appendix_c_resources)

* [ECMA 262 5th Edition December 2009](http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf)
* [Opera's The Web Standards Curriculum](http://dev.opera.com/articles/view/1-introduction-to-the-web-standards-cur/#toc) - basic articles about building with web standards.
* [10 Principles of the CSS Masters](http://nettuts.com/html-css-techniques/10-principles-of-the-css-masters/) - all of this is smart advice.
* [Google](http://www.google.com/) - when in doubt, google it.
* [Too lazy?](http://lmgtfy.com/?q=front-end+development+coding+standards&l=1) - let me google that for you then...
* [Yahoo's Exceptional Performance](http://developer.yahoo.com/performance/)team has maintained one of the best summaries of performance advice.
* [Google now has a Speed site](http://code.google.com/speed/page-speed/docs/rules_intro.html) with excellent detail. (Click All best practices).
* There are many more excellent resources on [page optimization](http://delicious.com/paul.irish/optimization) and [javascript performance](http://delicious.com/paul.irish/performance).
* [Mozilla's coding standards](http://wiki.mozilla.org/WebDev:FrontendCodeStandards).
* [Nokia's JavaScript Performance Best Practices](http://wiki.forum.nokia.com/index.php/JavaScript_Performance_Best_Practices).

**作者**

* [Paul Irish](http://twitter.com/#%21/paul_irish)
* [Nick Cooley](http://twitter.com/#%21/nickcooley)
* [Adam McIntyre](http://twitter.com/#%21/_ajm)
* [Rob Larsen](http://htmlcssjavascript.com/)
* [Joel Oliviera](http://twitter.com/#%21/jayroh)
* [Jared Williams](http://twitter.com/#%21/jaredwilli)
* [Annette Arabasz](http://twitter.com/#%21/annettish)
* [Rob Cherny](http://twitter.com/#%21/rcherny)

**修改历史**[◊](http://coderlmn.github.io/code-standards/#_revision_history)[**返回顶部**](http://coderlmn.github.io/code-standards/)

* 2012 Fall/Winter: Conversion to Github Hosted Version; general edits (larger edits coming soon!)
* 2011 May/June: Entire Standards page redesign. Coded with html5boilerplate.
* 2011 April: Web Typography document created.
* 2011 March: 2k11 clean up. Added contact info, etc. etc.
* 2010 April: Prep for public release.
* 2010 March: Merge with other documents and Fellowship's coding standards. HTML format.
* 2009 August: Performance Recommendation document created.
* 2008 July: [Reduce size of document](https://wiki.molecular.com/wiki/Interface_Development_Coding_Standards)
* 2008 July: [Merge with new standards](https://wiki.molecular.com/w/index.php?title=Coding_Standards&oldid=9956).
* 2005 April: Original "how to write good markup" document. [UID\_coding\_standards.doc](https://fusion.molecular.com/C14/Interface%20Development/Interface%20Development/UID_coding_standards.doc).